Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 25 – 32

Procedia Computer Science

3rd International Conference on Recent Trends in Computing (ICRTC 2015)

Multimedia Streaming using Cloud-Based P2P Systems

Navin Thomas, Mathew Thomas, K. Chandrasekaran

Department of Computer Science and Engineering, National Institute of Technology Karnataka, Surathkal, Mangalore

Abstract

The past one and a half decades have seen great strides in the field of commercially available distributed computing implementations. The two most popular architectures in the modern world are Peer-to-Peer Systems and Cloud Systems. Peer-to-Peer Systems (P2P) have become very popular in recent times, mainly being used to facilitate file sharing among disparate systems. Another recent trend in modern computing has been the wide scale utilization of Cloud Computing architectures. These systems are used to allow multiple systems to pool their resources and allow other tertiary systems to use these shared resources in bulk for tasks such as data storage, complex calculations, and file sharing. This entails the conceptual outsourcing of various data processing tasks to an external cloud system. Given the two nearly independent functionalities of P2P and Cloud architectures, it is interesting to consider the possibility of fusing these two concepts and researching the applications of the resultant amalgamation.

In this research paper, we discuss the theory and application of Cloud Based Peer-to-Peer Systems and their potential application in multimedia streaming services. While the value of P2P Systems and Cloud Computing Systems have been extolled individually, the hybrid of both concepts shows great promise. In this paper we provide an introduction to Cloud Computing Systems, P2P Systems, and the advantages as well as the limitations of both configurations. We then describe the concept of a Cloud-Based P2P System, its basic architecture, and its possible implementations. We also describe the possible application of a Cloud-Based P2P System as a platform for a multimedia streaming service. A proposed algorithm to facilitate streaming in such an application is also described, along with a proposed system model and its advantages.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Cloud Computing; Peer-to-Peer Computing; Multimedia Streaming;

1. Introduction

Cloud Computing involves joining a large number of remote systems connected over a network and utilizing their hardware and software resources to share tasks and data over the internet. These combined system resources are shared by multiple users and allocated according to the demand of each individual cloud client. Such a service

* Corresponding author. Tel.: +91-824-2473400 ; Fax: +91-824-2474033.
E-mail address: kchnitk@ieee.org

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.359

allows the client to outsource their hardware and software needs to cloud service providers. This thereby allows them to avoid hefty upfront infrastructure costs. The different types of cloud computing deployment models [1] include:

- ***Private***: Private clouds are set up for use by a single organization. They are either managed internally or by third-party service providers, and hosted internally or externally. Such a model provides the organization with an increased level of security and privacy, if implemented properly.
  Setting up a private cloud requires a considerable amount of effort to make sure that any security issues or vulnerabilities that arise during setup are resolved. However, the end product provides the organization with much more control over their network and resources.
- ***Public***: In Public clouds, cloud services are offered to the general public over large networks, such as the Internet. So each client has access to the same infrastructure and services, providing a uniformity of access for all users. Common examples of Public cloud service include Google Drive and iCloud.
  Public clouds require a very large number of servers to be connected to form the cloud in order to be useful for the targeted wide user base. As a result, a properly constructed Public cloud ensures reliability due to the vast number of interconnected systems and networks involved. This provides a very high fault tolerance, since there is always a backup system ready to take over for a failed system. This also provides an ideal scalability factor, allowing resources to be allocated as and when required to clients with variable priority. If the Public cloud is accessed through the internet, it can provide location independence to their users as well.
- ***Hybrid***: Hybrid clouds utilize a combination of Private and Public models to provide services to the same organization. The logic behind this is that services which do not require high levels of security can be managed using Public cloud methodology, while all secure operations and resources can be utilized with Private cloud methodology. This is done because Public cloud services are much more cost efficient than Private cloud services, since we do not have to take security and privacy constraints into consideration to such a large extent.

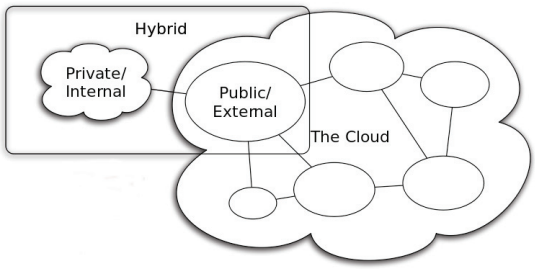

Fig. 1. Different categories of Cloud models

Despite advantages such as decreased cost, location independence, and device independence, there is still some speculation regarding the security of cloud based systems (particularly Public clouds), due to the remote nature of data storage.

Peer-to-peer (P2P) Computing also facilitates the sharing of tasks and data between interconnected systems (peers). However, unlike Cloud Computing, peers are all hierarchically equal and they must all perform the same tasks (albeit at different instances). For example, in a P2P file sharing protocol, such as BitTorrent, peers both provide and use resources (i.e. share files and obtain files, sometimes simultaneously)[2]. This is fundamentally different from both cloud and client-server models, where only certain systems provide services. We can better define P2P systems as distributed systems which satisfy the following properties[3] :

- ***High Degree of Decentralization***: Each peer performs the same services, i.e., they act as both server and client to each other. Hitherto, all the system resources such as bandwidth, storage, and processing power are provided jointly by all the participating peers.
- ***Self-Organization*** : Nodes can be easily inserted into the system. This means that little or no configuration or setup is required to add peers to the system, or maintain them.
- ***Multiple Administrative Domains***: There is no single entity or organization which controls all the peers in the network. Usually each peer is owned by a separate individual who connects to the network voluntarily to take advantage of its services.
- ***Low Initial Investment***: Since P2P systems are mainly composed of the hardware resources of individual peers, which are supplied by the peers themselves, there is very little need for initial investment before deploying a P2P network.
- ***Organic Growth***: Available resources are dependent almost entirely on the external peers which connect to the network. Therefore, P2P systems have the potential to grow almost limitlessly, without the need of any large hardware upgrade, as would normally be required for the expansion of a standard client-server type system.
- ***High Fault Tolerance***: Since a P2P system is highly decentralized and all peers are equal hierarchically, fault tolerance should be comparatively very high. No peer has a higher importance than any other peer, so at the occurrence of a system failure, there would be many systems ready to take its place.
- ***Abundance and Diversity of Resources***: A large number of diverse systems can be connected together as peers in a P2P system. Therefore, a good P2P system supplies a large variety of different and powerful resources to any single peer at a fraction of what it would cost to obtain those components individually.

These are a few general characteristics and advantages of a good P2P system. Researchers use P2P systems to share large amounts of data, such as software and updates, easily and quickly. Distributed data processing can also be facilitated using P2P systems.
However there are still concerns regarding the security of P2P systems, manageability of exceedingly large and decentralized P2P systems, as well as the use of P2P systems to enable illegal distribution of media.

Cloud-Based Peer-to-Peer Systems : Having seen the merits of both Cloud systems and P2P systems individually, we can now look at the reasons to combine the two. One of the chief disadvantages of Cloud systems is the centralization of a particular service. If the cloud server were to fail, the system would fail, therefore making the system less fault tolerant. The obvious solution is to provide backup cloud servers to take over in case of a system failure. The backup systems should be equivalent, so that it can successfully undertake the tasks of the erroneous component. As we can see, a P2P system architecture is ideal in this case, as peers have equal hierarchical status.
A Cloud Based P2P System also provides a greater level of scalability. In order to upgrade a cloud system, we would have to make a large investment. However, with a P2P arrangement, the system can be improved by adding peers, since the processing power and resources of a P2P system are determined by the peers comprising the system.
Therefore, while a traditional centralized Cloud system certainly has its merits, for making a Cloud service available to a larger, more widespread audience, a Cloud Based Peer-to-Peer System is ideal. This architecture would provide an efficient method of access for a large as well as geographically diverse user base.

2. Related Work

Babaglou, Marzolla, and Tamburini[4] show us the benefits of utilizing a Cloud-Based P2P System instead of other system models. However, there is no reason to implement such a complicated system unless there is a definite and tangible application. Streaming of high quality audio and video has always been a challenge due to storage and bandwidth constraints. We can see that the use of a Cloud-Based P2P System is ideally suited for this situation.
CLive[5] is one of the applications which attempts to use the advantages of Cloud-Based P2P Systems to provide a video streaming service with high QoS. Another existing application is the Novasky[6] service which has been deployed in Tsinghua University. Novasky provides bit rates of 1-2 Mbps while managing over 1000 high quality video streams. This gave us the idea of determining an algorithm to find the nearest peer which is capable of streaming the desired content, thereby decreasing latency even further.
While research has been done on various other system models and architectures[7], we have devised a model that we

believe should ideally suit the needs of the client while ensuring system manageability. Such a system model can be utilized by both small scale and very large scale users, as the system allows the interconnection between both types of systems. The utility of a Cloud-Based P2P System was demonstrated by Ke Xu et al.[8], with the virtues of decentralization being clearly described. The next step in this application's development would involve accessing the Cloud-Based P2P System via a mobile device, such as a smartphone. This has been analyzed thoroughly in the development of StreamSmart[9].

3. System Model

The essence of the proposed system model consists of a network of interconnected cloud systems. These individually owned autonomous cloud systems are completely self sufficient and capable of providing cloud services to their respective clientele without the aid of other nodes in the network. However, each cloud system joins the network to take advantage of the shared resources of the entire system. Naturally, this is accomplished by giving each of these cloud systems a hierarchically equal level and equal access to all other connected nodes. This, by definition, is a description of a P2P System. Therefore, by fusing the architectures of cloud based and P2P systems, we establish a very powerful and resource rich hybrid architecture in the form of a Cloud-Based P2P System[4].

Each node provides and shares certain resources to the system as a whole. These shared resources include processing elements, primary and/or secondary storage, and connectivity. The entire aim of forming this interconnected system is to share these resources. To do this, there needs to be two types of communication these are node-to-node communication and user-to-node communication, (i.e., an interface between a client and an individual cloud system, by which the user may provide input and receive output). This would be facilitated by installing a software daemon on each node. This mandatory software daemon is split into two sections, one for the inter-node communication, and one to provide a suitable user interface. The daemon also efficiently manages network churn (the sudden arrival or departure of peers) and maintains a minimum level of cohesion among the different nodes.

To allow the Cloud-Based P2P System to share resources in an ideal manner, the system resources can be divided and assigned to tasks according to the hardware requirements of each individual task. A user makes a request for a certain operation or task to be performed. The system checks whether it has enough unallocated resources to execute the impending operation. If the available resources are sufficient, then a partition or slice is allocated to the task and the operation is carried out. If enough system resources are not available, then the client node is made to wait until resources are freed.

The system slices allocated to individual tasks are dynamic in nature and can be varied based on user input. Supposing a user wants to decide how much of the system resources are allocated for their task, they can shrink or augment their allocated slice, provided their changes do not affect the slices of other tasks.

Since we have stipulated that each node is an autonomous cloud system, it is apparent that there cannot be any set guarantee of Quality of Service. Results may vary due to system churn and node failure. However the Cloud-Based P2P System ensures that there is a good level of cohesion among the different slices. This means that even in the event of multiple node failures, the surviving nodes can still take over the load of the failed systems. Simply put, if we consider the Cloud-Based P2P System to be an actual cloud, even if a few droplets join or fall away, the cloud still maintains its overall shape.

3.1. Advantages of Proposed System Model

3.1.1. Fault Tolerance

The system model described above has a high resilience to errors and system failures. Such a high level of fault tolerance is required in order to accommodate the autonomous nature of each respective node. This means that each node is not expected to be reliable, i.e., they may connect and disconnect from the network at will. Accordingly, once a slice of the entire Cloud-Based P2P Network is allocated for a particular task, there has to be a set of backup systems available to take over in case a particular node within that slice disconnects from the network. Hitherto, fault tolerance is not only ensured, it is required to maintain the integrity of the network[10].

3.1.2. Scalability

Scalability in a cloud network is always an obstacle, since the hardware required to successfully upgrade a highly centralized cloud is very expensive. Thus the incorporation of Peer-to-Peer methodology is an optimal solution to this limitation. By allowing already pre-existing clouds to connect together and pool their resources, we cut the cost of upgrading drastically, and still maintain independence from other cloud systems[11].

3.1.3. Resource Allocation

In any distributed system, efficient resource allocation methods are a key factor. In our Cloud-Based P2P System model, successful resource allocation manifests itself in the form of the allocation of slices. Depending on the level of processing power and memory required for a particular task, a node may request an appropriate slice of the entire system's resources. The system then determines whether the desired task can be accomplished using available resources. If the available resources are sufficient, an adequate slice of system resources are deployed for the task, and the operation is carried out. If not, the task is made to wait until enough resources are available[11].

4. Application in Multimedia Streaming

There are multiple uses for Cloud-Based P2P Systems, such as wide area cloud storage services[12], and various gaming applications[13]. We have chosen to explore the possible applications of Cloud-Based P2P System in streaming services.

P2P streaming services have been emerging as of late, but have faced a setback when confronted with the bottleneck that arises due to bandwidth restrictions. These restrictions can greatly hamper the Quality of Service (QoS), especially when dealing with high fidelity data such as HD video or lossless audio recordings[5]. Using a cloud service, we can overcome this by allowing the cloud to assist the P2P system. This means we can maintain a certain threshold level of QoS.

In such an approach, temporary cloud resources (helpers) are supplied as required to the system. This increases the amount of available bandwidth and thereby improves the chances of receiving the requested multimedia on time. This constitutes the basic principle of a hybrid Cloud-Based P2P System.

5. Prototype Design

One of the main reasons for using an Cloud-Based P2P system is to provide access to local nodes (in this case, the nearest Cloud server). So the client should be able to search for his/her desired content and stream it from the nearest node with the highest available bandwidth so as to obtain best QoS.

Therefore, we have to develop a method of connection establishment and efficient streaming to provide the fastest possible data transfer rates for smooth streaming.We have formulated a general process to provide a viable streaming service. It is described below.

1. Establish a connection from the client device to the nearest node, i.e., cloud server. This nearest node is referred to as the "Primary Node"
2. Specify the desired content to be streamed
3. The Primary Node determines the shortest path (among the various possible paths) to a node containing the desired content. Here, shortest path is determined by two parameters, bandwidth and distance to the host node. These parameters are judged by measuring Round-Trip Time (RTT). The Primary Node uses a process involving Dijkstra's Algorithm[14] to compare the RTTs of different paths and determine the ideal path
4. Content is streamed to the client device from the host node via the Primary Node.
5. Connection is terminated

The following algorithm shows the method by which the Primary Node finds the shortest path to a node containing the requested file. It involves the use of Dijkstra's Algorithm. However, we must keep in mind that Dijkstra's Algorithm is only used when a node is added or removed from the network. Otherwise, each node stores a list containing the shortest path to every other node. The algorithm is described below.

Algorithm 1 : Multimedia Streaming on a Cloud based P2P System using Dijkstra's Algorithm

---

// To stream multimedia file from the nearest node on the // P2P network

Given a list of nodes $N : n_1, n_2, ...n_n$ in a P2P network, a list of multimedia files V_i and empty list D_i, E_i corresponding to node n_i;

if a node is added or removed from the network **then**
  for all $n \in N$ **do**

    Find paths between node n and its neighbouring nodes;

  end for

  for all $n \in N$ **do**

    Find shortest path between node n and rest of the nodes on the P2P network using Dijkstra's Algorithm and update the list D_i;

  end for
end if

if a node n_i wants to stream a particular multimedia file v **then**

  node n_i sends a broadcast message to all other nodes to check if the node has the multimedia file in its cloud storage;

  for all $n \in N - n_i$ **do**

    if $v \in V_n$ **then**

      node n sends acknowledgement to node n_i

      node n_i adds node n to its list E_i

    end if
  end for

  Using lists D_i and E_i, find the node in list E_i with the shortest path say node p;
  Stream multimedia file from node p;

end if

As shown in the algorithm, determination of an optimal path depends entirely on the use of Dijkstra's Shortest Path Algorithm. Therefore it is important to understand its working.

5.1. Dijkstra's Algorithm

For a given graph, Dijkstra's algorithm finds the path with lowest cost (i.e. least RTT) between a source node and every other node. Using this algorithm, we can find the lowest cost between each and every node.

5.1.1. Principle of Dijkstra's Algorithm

Consider a network $G = (V, E, w)$ having non-negative weights and $V = (v_1, v_2, ..., v_n)$. Then the minimum $G_{(v_i,v_j)} \in E$ will satisfy the following function(assuming c_1 is the source node):

$$c_1 = 0 \tag{1}$$

$$c_j = min(c_j, c_i + w_{ij}) \tag{2}$$

where c_j is the cost of node v_j

5.1.2. Asymptotic Analysis

By storing the vertices of a set Q in a linked list or array, we can easily extract minimum from Q by linear searching using Dijkstra's algorithm. From this implementation, the running time is $O(|E| + |V|^2) = O(|V|^2)$.

In cases where the graphs have much less than $O(|V|^2)$ edges, for example, in the case of sparse graphs, we can implement Dijsktra's algorithm much more efficiently by storing the graph in the form of an adjacency list. We can use self-balancing binary search trees, binary heaps, pairing heaps, or Fibonacci heaps as priority queues are used to facilitate the extraction of the minimum in an efficient manner. The running time of Dijkstra's Algorithm with a self-balancing binary search tree or a binary heap is $\Theta((|E|+|V|)\log|V|)$ in its worst case (also $Ø(|E|\log|V|)$, assuming the graph is connected). The use of a Fibonacci heap (Fig. 2) improves this to $O(|E| + |V|\log|V|)$.

5.1.3. Graphical Analysis

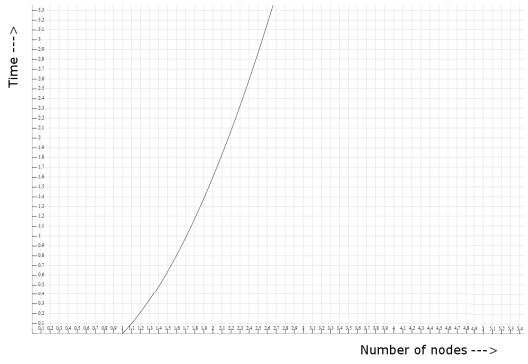

Fig. 2. Best Case : Fibonacci Heaps

6. Conclusion

In this paper we have described Cloud Systems, P2P Systems, and their different advantages, disadvantages, and classifications. Keeping this information in mind, we introduced the hybrid model of a Cloud-Based P2P System and its improvements on the other two systems.

We further developed the idea of an implementation of the Cloud-Based P2P System for a multimedia streaming application. This involved creating an algorithm for communication between the client and nodes, and also between different nodes, while providing the fastest streaming service with highest QoS. The proposed algorithm involved the use of Dijkstra's Algorithm which we described in detail using asymptotic and graphical analysis.

Further endeavours will include improvement on the existing algorithm, as well as the addition of a method to govern fault tolerance (while the system is resilient to faults, an algorithm is required to determine the redistribution of system resources in a slice after a single node fails). Another possible improvement involves the addition of dynamic relocation of data based on demand. This would entail the transfer of files which are in high demand by the clients of a particular node, being transferred to that node for faster access. An algorithm would be required to determine at what threshold value of demand such a transfer would take place. Full implementation would require testing on large scale systems, in order to verify the application's viability in a mass market.

References

1. *What is Cloud Computing : Public Clouds, Private Clouds, and Hybrid Clouds*, URL. http://www.interoute.com/cloud-article/what-cloud-computing
2. *Wikipedia : Peer-to-Peer*, URL. http://en.wikipedia.org/wiki/Peer-to-peer
3. *Rodrigo Rodrigues, Peter Druschel - Peer-to-Peer Systems*, Communications of the ACM, Vol. 53 No. 10, Pages 72-82 10.1145/1831407.1831427 http://cacm.acm.org/magazines/2010/10/99498-peer-to-peer-systems/fulltext
4. *Babaoglu, Marzolla, and Tamburini - Design and Implementation of a P2P Cloud System*, SAC '12 Proceedings of the 27th Annual ACM Symposium on Applied Computing. Pages 412-417
5. *Amir H. Payberah, Hanna Kavalionak, Vimalkumar Kumaresan, Alberto Montresor, Seif Haridi - CLive: Cloud-Assisted P2P Live Streaming*, Swedish Institute of Computer Science (SICS), Sweden University of Trento, Italy. IEEE P2P 2012 Proceedings.
6. *Fangming Liu, Shijun Shen, Bo Li, Baochun Li, Hao Yin, Sanli Li - Novasky: Cinematic-Quality VoD in a P2P Storage Cloud*, Hong Kong University of Science & Technology, Tsinghua University, University of Toronto. IEEE INFOCOM 2011.
7. *Irena Trajkovska, Joaqun Salvacha Rodrgues, Alberto Mozo Velasco - A Novel P2P and Cloud Computing Hybrid Architecture for Multimedia Streaming with QoS Cost Functions*, Technical University of Madrid, Spain.
8. *Ke Xu , Meina Song , Xiaoqi Zhang , Junde Song - A Cloud Computing Platform Based on P2P*, Beijing University of Posts and Telecommunications, Beijing, China
9. *Alessandro Gaeta, Sokol Kosta, Julinda Stefa, and Alessandro Mei - StreamSmart: P2P Video Streaming for Smartphones Through The Cloud*, Department of Computer Science, Sapienza University of Rome, Italy. IEEE SECON 2013
10. *Xiaofei Zhang and Lei Chen - Fault Tolerance Study for Durable Storage on the Cloud*, Department of Computer Science and Engineering,HKUST,Clear Water Bay, Kowloon, Hong Kong. 2011 International Conference on Cloud and Service Computing.
11. *Han Xingye, Li Xinming, and Liu Yinpeng - Research on Resource Management for Cloud Computing Based Information System*, 2010 International Conference on Computational and Information Sciences.
12. *Antonio Davoli and Alessandro Mei - Triton: a Peer-Assisted Cloud Storage System*, Computer Science Department Sapienza University of Rome.
13. *Richard Suselbeck, Gregor Schiele, and Christian Becker - Peer-to-Peer Support for Low-Latency Massively Multiplayer Online Games in the Cloud*, Universitat Mannheim, Germany. NetGames '09 Proceedings of the 8th Annual Workshop on Network and Systems Support for Games Article No. 14 IEEE Press Piscataway, NJ, USA 2009
14. *Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford - Introduction to Algorithms (Second ed.), "Section 24.3: Dijkstra's algorithm"*, (2001). MIT Press and McGrawHill. pp. 595601. ISBN 0-262-03293-7.