A LOAD BALANCING ALGORITHM BASED ON MOVEMENT OF NODE DATA FOR DYNAMIC STRUCTURED P2P SYSTEMS

1Prof. Prerna Kulkarni, 2Amey Tawade, 3Vinit Rane, 4Ashish Kumar Singh
1Asst. Professor, 2,3,4BE Student, 1,2,3,4Comp. Engg. Dept, SSJ CET, Asangaon, India.
1prernakulkarni09@gmail.com, 2ameeyta@gmail.com, 3vinitrane14@gmail.com, 4ashu20121994@gmail.com

Abstract - Load balancing is one of main challenge in day to day life. Load balancing is the balancing load between multiple peers to make system stable. In this paper, proposing load balancing algorithm for peer to peer file sharing system. In this, we are improving efficiency of system by reducing buffer size of data, response time and also depend on downtime and proximity. In this, we are proposing load balancing algorithm using node movement technique, while balancing load if node gets overloaded then the data will be transfer to another system using node movement technique. We can improve response time and efficiency by proximity and downtime.

Keywords — dynamicP2P, LoadBalancing, NodeMovement, DHT, StructuredSystem.

I. INTRODUCTION

Distribution of data tem the nodes in must structured peer to peer (P2P). System is done through data node movement. The mechanism that uses file sharing method among the node using this method. In the previous paper, we do load balancing using Distributed hash Table. In this approach load balancing done by hashing the key-space to the navigation space using a pseudo random hash function with uniform such functions destroys the locally property of data. This approach is hardly applicable to load balancing. Over node of node may be caused by huge amount of data stored at node. One of operation suitable in such condition node movement of data which are less located node. Load balancing is the major issue while sharing the system. Our aim to reduce the load on the node. Load balancing depend upon on many factor, while designing any file sharing mechanism. Balancing implies that load has to equalized rather than just shared. Load balancing attempt to maximize the response time.

Composition of load balancing consists of following factor:

1) Transfer policy: in this policy, we determine the state of node. Nodes are at start ideal condition. No node movement at ideal condition.

2) Selection policy: In this policy, we determine the proximity and downtime of the node.

Node movement is carried out in following condition:
1. A node which are getting heavily overloaded.
2. It also depends on the proximity and downtime of the node.
3. A node which are getting full capacity, then in this case node movement carried out.

PEER-TO-PEER (P2P) systems have emerged as an appealing solution for sharing and locating resources over the Internet. Several P2P systems have been successfully deployed for a wide range of applications. The basic approach to load balancing is to find a pair of nodes—one that is heavily loaded and the other lightly loaded—and redistribute the load across these two nodes. However, it is far from trivial to (globally) balance the load in a P2P system. There are two main issues in P2P’s load balancing: 1) how to determine if a node is overloaded or under loaded, and 2) An important problem is to decide how to achieve a balance in the load distribution between processors so that the computation is completed in the shortest possible time. [7] A popular solution is to let each node in the system query for the load of an arbitrary
number of other nodes periodically. If the number of
queried nodes is large enough, the node can approximate
the average load of the system, and hence, it can determine
if it is overloaded or under loaded. If the node is overloaded
(or under loaded), it redistributes its load with the queried
node having the lightest (or heaviest) load since that node
should be a lightly (or heavily) loaded node. Distributed
hash tables provide a solution to the lookup problem in
distributed systems. Given the name of a data item stored
somewhere in the system, the DHT can determine the node
on which that data item should be stored, often with time
complexity logarithmic in the size of the network. There
are two main goals to be achieved, minimize the load
balance and minimize the amount of load moved. If the hot
peers become bottlenecks, it leads to increased user response
time and significant performance degradation of the system.
Hence the load balancing mechanism is necessary in such
cases. With the notion of virtual servers, peers participating
in a heterogeneous, structured peer-to-peer (P2P) network
may host different numbers of virtual servers, and by
migrating virtual servers, peers can balance their loads
proportional to their capacities. The security vulnerabilities
are analyzed of the typical DHT load balancing mechanism;
then propose an algorithm that both facilitates good
performance and does not dilute security.

Types of Load Balancing Algorithms:

Load balancing algorithms can have three categories based
on initiation of process as follows:

- **Sender Initiated:** In this type the load balancing
  algorithm is initialized by the sender. In this type of
  algorithm the sender sends request messages till it finds
  a receiver that can accept the load.

- **Receiver Initiated:** In this type the load balancing
  algorithm is initiated by the receiver. In this type of
  description algorithms the receiver sends request messages
  till it finds a sender that can get the load.

- **Symmetric:** It is the combination of both sender initiated
  and receiver initiated [1].

## II. LITERATURE SURVEY

### Types of P2P Networks:

P2P is a paradigm for sharing of computing
resources/services such as data files, cache storage, and disk
space or processing cycles. In comparison with the
conventional client/server model, P2P systems are
characterized by symmetric roles among the peers, where
every node in the network acts alike and the processing and
communication are widely distributed among the peers.
Unlike the conventional centralized systems, P2P systems
offer scalability and fault-tolerance. It is a feasible
approach to implement global-scale systems such as the
Grid. An important achievement of P2P networks is that all
clients provide resources, including bandwidth, storage
space, and computing power. Thus, as nodes arrive and
demand on the system increases, the total capacity of the
system also increases. This is not true for client/server
architecture with a fixed set of servers, in which adding
more clients could mean slower data transfer for all users.
The distributed nature of P2P networks also increases
robustness in case of failures by replicating data over
multiple peers, and in pure P2P systems by enabling peers
to find the data without relying on a centralized index
server. In the latter case, there is no single point of failure
in the system.

### Structured P2P (P2P) Networks:

Structured P2P network employ a globally consistent
protocol to ensure that any node can efficiently route a
search to some peer that has the desired file, even if the file
is extremely rare. Such a guarantee necessitates a more
structured pattern of overlay links. By far the most common
type of structured P2P network is the distributed hash table
(DHT), in which a variant of consistent hashing is used to
assign ownership of each file. The ID management
algorithm presented here is a greedy distributed algorithm
that directs joining peers to highly-frequented regions of the
ID space [5].

### Unstructured Peer to Peer Networks:

An unstructured P2P network is formed when the overlay
links are established arbitrarily. Such networks can be
easily constructed as a new peer that wants to join the
network can copy existing links of another node and then
form its own links over time. In an unstructured P2P
network, if a peer wants to find a desired piece of data in
the network, the query has to be flooded through the
network to find as many peers as possible that share the
data. The main disadvantage with such networks is that the
queries may not always be resolved. But if a peer is looking
for rare data shared by only a few other peers, then it is
highly unlikely that search will be successful. Since there is
no correlation between a peer and the content managed by
it, there is no guarantee that flooding will find a peer that
has the desired data. Flooding also causes a high amount of
signaling traffic in the network and hence such networks
typically have very poor search efficiency.
Let's take analysis of different developed peer to peer technology for efficient load balancing results and our proposed algorithm fastest load balancing results. Different load balancing algorithms applicable efficient peer to peer load balancing results.

Various popular load balancing algorithms comparison given as follows:

<table>
<thead>
<tr>
<th>Table 1: Difference between all load balancing algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameters</td>
</tr>
<tr>
<td>Nature</td>
</tr>
<tr>
<td>Overload rejection</td>
</tr>
<tr>
<td>Reliability</td>
</tr>
<tr>
<td>Stability</td>
</tr>
<tr>
<td>Fault tolerant</td>
</tr>
<tr>
<td>Resource utilization</td>
</tr>
<tr>
<td>Response time</td>
</tr>
<tr>
<td>Waiting time</td>
</tr>
<tr>
<td>Turnaround time</td>
</tr>
<tr>
<td>Execution system</td>
</tr>
<tr>
<td>Throughput</td>
</tr>
<tr>
<td>Processor thrashing</td>
</tr>
<tr>
<td>Predictability</td>
</tr>
<tr>
<td>Adaptability</td>
</tr>
</tbody>
</table>

III. PROPOSED SYSTEM

Stages in proposed system:
- Select the target node.
- Select the data from target node.
- Send to the receiver node.
- If the receiver data storage is full then it selects nearest and time efficient node. Send the data to new receiver node.

IV. SYSTEM ARCHITECTURE

![System Architecture Diagram]

V. MATHEMATICAL MODEL

While minimizing load movement algorithm try to minimize load imbalance factor. Some other important factors which are related to the destination of the load transfer also considered. The cost of transferring load to a destination node is based on destination load, downtime and its proximity to the overloaded node. by using the following formulas want to select nodes in the related group that returns the minimum cost. So when to move some load from a node q to a node p the destination cost is formulated as below:

\[ \text{DestinationCost} = w_1 \times \text{Load\_status}_p + w_2 \times (\text{loc}_q - \text{loc}_p) / \text{distance}\_\text{max} + w_3 \times (\text{downtime}_p / t) \]  

In (1), cap and loc denote the capacity and location of a node respectively. To normalize the location parameter in (1), divide the result of subtracting locations by distance\_\text{max} that stands for the distance between i and the farthest node in the related group. The load of each object k is defined as follows:

\[ \text{Load\_k} = \text{size} \times r \]
In formula (3), calculate the average amount of bytes that is transferred in each unit of time in relation to object k. supposing that there are r requests for the object k in the related time unit, average they sent bytes for these r requests and set the parameter size to the achieved result.

VI. PROPOSED ALGORITHM

Node Movement:

Node movement is done when one of the following cases arises:

1. A node gets overloaded due to the high popularity of more than one of its items.
2. A node gets overloaded because of high amount of data items put on it while none of them is highly popular. In this case the popular-item-list for this is empty.
3. A node gets overloaded while there is only one item in its popular-item-list. This item key is not equal to node’s key and also the node capacity is less than half of the average node’s capacities in the related directory.

‘Pushing’ non-hot data (via migration for large-sized data and via replication for small-sized data) to large capacity peers as much as possible.

Also when a node gets overloaded due to excess number of assigned items to it while none of them is highly popular, it apply node movement to move some of these items to other nodes. For the case that there is only one popular item in an overloaded node’s popular-item-list, it improbable that due to its increasing popularity rate, moving this item to another node causes that node to get overloaded too. But considering system heterogeneity, it is possible that this node’s overloading be much more due to its low capacity and not because of the great number of requests for the so-called popular item. So to delay replication, it balances the load of nodes even in this case by node movement if it is possible. To this end, use nodes’ capacities information that is stored in each directory to estimate the average capacity of nodes in the system (Cavg).

Suppose than n is the overload node and all node’s flags are not set.

VII. EXPECTED RESULTS

![Fig.1 system optimization graph](image-url)
REFERENCE


VIII. CONCLUSION

In this paper, propose a load balancing algorithm for peer to peer file sharing system. In this each node balances the load by using the concept of node movement. In this we also use the concept of proximity and downtime to enhance efficiency of system. It also increases the response time of system. So the work done will be faster. In this we have also worked on buffer size of data so that it will be easy to share data efficiently.