IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Histogram Based Live Streaming in Peer to Peer Dynamic Balancing & Clustering System

M.Kavita[*1], Ms. P. Thamarai[2] Dr. T.V.U. KiranKumar[3]

[*1,2,3] Department of Electronics and Communication Engineering, Bharath University, Selaiyur, Chennai - 600 073, Tamilnadu, India

kavita.mani06@gmail.com

Abstract

We made a preliminary clustering analysis of an investigational peer-to-peer system, tested in a small measure research within Planet Lab. The application was an Internet-TV like streaming system based on Chord architecture. Our clustering is inspired by the Regularity. Such approach was already demonstrated in biology and appeared to be a powerful tool. Live streaming result suggests that the nodes of a large enough graph can be partitioned in few clusters in such a way that link distribution between most of the pairs look like random. Our main goal is to study what this type of clustering can tell us about p2p systems using our investigational system as source of data. We searched clustering's of clustering type by using max like hood as leadership. Our graph is directed and weighted. The link direction designates a client server relation and the value is the proportion of all chunks obtained from such a link during the whole experimentation. We think that the first results are motivating. Most of the cluster pairs have very great patterns of link distribution, demonstrating that such a novel approach has potential in organizing peers effectively. The values of weights between clusters and their distribution show some apparent patterns. We end up with cluster pairs

Keyword: Load balancing, histgrom, clustering.

Introduction

There is a developing market for IPTV. Various commercial systems now offer services over the Internet that is similar to traditional over-the-air, cable, or satellite TV. Live small screen, time-shifted programming, and content-on request are all presently available over the Internet. Improved broadband speed, growth of broadband contribution base, and improved video compression technologies have contributed to the emergence of these IPTV services. We draw a distinction between three uses of peer-to-peer (P2P) networks: delay tolerant file download of archival material, delay sensitive progressive download (or streaming) of archival material, and present live streaming.

In the first case, the completion of download is elastic, contingent on available bandwidth in the P2P network. The submission buffer receives data as it trickles in and informs the user upon the completion of download. Bit torrent and variants are example of delay-tolerant file download systems. In the second case, video repetition starts as soon as the application assesses it has sufficient data buffered that, given the predictable download rate and the playback rate, it will not reduce the buffer before the end of file.

If this assessment is wrong, the application would have to either pause playback or rebuffed, or slow down playback. While users would like playback to start as soon as possible, the application has some step of freedom in trading off playback start time against estimated network capacity. The third case, real-time live streaming has the most stringent delay requirement. While advanced download may tolerate initial buffering of tens of seconds or even minutes, live streaming normally cannot tolerate more than a few seconds of buffering.

Taking into account the delay introduced by signal ingest and encoding, and network transmission and broadcast, the live streaming system can introduce only a few seconds of buffering time end-to-end and still be considered "live"

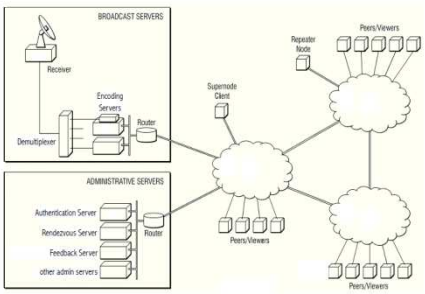


Fig 1: System Architecture

Overlay Network

An overlay network is a virtual computer network that is built on top of another network. Nodes in the overlay remain connected by virtual or logical links, every of which corresponds to a path, perhaps through countless physical links, in the primary network. The topology of the overlay network may (and often does) differ from that of the underlying one.

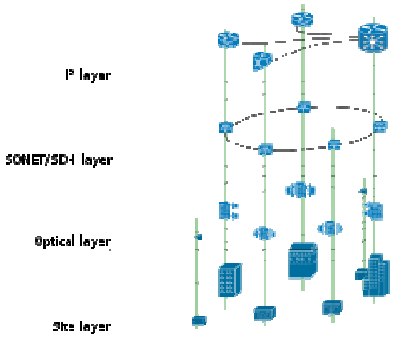


Fig 2: Overlay Network

For example, many peer-to-peer networks are overlay networks because they are organized as nodes of a virtual system of links run on top of the Internet. The Internet was originally built as an overlay on the telephone network .[14]

The most striking example of an overlay web, however, is the Internet itself: At the IP layer, each node can reach any added by a direct connection to the desired IP address, thereby creating a fully linked network; the underlying network, however, is self-possessed of a mesh-like interconnect of sub networks of varying topologies (and, in fact, technologies). Address resolution and direction-finding are the means which allows the mapping of the fully-connected IP overlay network to the underlying ones. Overlay networks have been everywhere since the invention of networking when computer systems were connected over telephone lines using modems, before any data network existed. Another example of an overlay network is a distributed hash table, which maps keys to nodes in the web. In this case, the underlying network is an IP network, and the overlay network is a table (actually map) indexed by keys.

Overlay networks have also been proposed as a way to improve Internet direction-finding, such as through quality of service guarantees to achieve higher-quality streaming media. Previous applications such as IntServ, DiffServ, and IP Multicast require not seen wide acceptance largely because they require modification of all routers in the network. On the other hand, an connection network can be incrementally deployed on end-hosts running the overlay protocol software, without support from Internet service providers. The overlap has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for illustration, the sequence of overlay nodes a message traverses before reaching its destination.

The Higlob Framework

In this section, we present the HiGLOB outline. We attention on the histogram and load-balancing managers.

The Histogram Manager The objective of the histogram manager is to maintain statistics about the load distribution across the entire P2P network. These figures allow a node to know its own load status (in comparison with other nodes in the system) and to identify its counterpart if global load balancing is triggered.

Histogram Structure

In our framework, each peer node P stores an estimated histogram, keeping the load distribution of the organization. The histogram contains several buckets, each of which keeps arithmetical information about the load of a group of nodes that is connected to P through a neighbor node. The statistical material includes the current workload of CPU, storing, and bandwidth of nodes in the group. Furthermore, to balance the load of heterogeneous nodes, the arithmetical information also contains a summary of the available resources of the group (also in terms of CPU, storage, and bandwidth). The weight of a node or group is calculated as the ratio between the current workload and the capability of the node or group.1 As a product, from the histogram info, the system can balance the load of nodes conferring to any one of a variety of node experiences storage, CPU, or bandwidth.

Model Description

Our task is to cluster nodes of a p2p system with $n = 48$ nodes. The experiment run for a time that corresponds to streaming of approximately 4000 chunks and all except the seed get almost the same

number of chunks during this period. The total (integral) number of chunks obtained by peer i is denoted as ni. By ni, we denote the integral number of chunks peer number i downloaded from peer number j. Then we define weighted graph by defining the weights for all pairs (i, j), $i \neq j$:

$$w_{i,j} = \frac{n_{i,j}}{n_i}.$$

Thus $0 \leq wi, j \leq 1$. With this weighted and directed graph we associate a directed random graph $\mathcal{G}w$ with independent links having probabilities.
$P((i, j) \in Ew) = wi,j$,

Which together with the link independence assumption defines the probability space. The reason why we defined $\mathcal{G}w$ is technical. We assume that if the original graph has a Sz. structure, then $\mathcal{G}w$ should have similar structure as well. A benefit is that for $\mathcal{G}w$ we can use max likelihood fitting methods that allow computations. This results in some approximate scheme that is difficult to evaluate exactly. However, our guidance is mainly the result that we get, if the clusters are meaningful, the method is positively working. To define the Sz. Structure framework we need two partitions $\mathcal{U} = \{U1, U2, \cdots, UKout\}$ and $\mathcal{V} = \{V1, V2, \cdots, VKin\}$ of node set $V : V = U1 + U2 + \cdots + UKout$ and $V = V1 + V2 + \cdots + VKin$, see in [7]. $\mathcal{U}$ is called out-group, and $\mathcal{V}$ in-group. The Sz. structure in this case means that we have 'regularity' in link weight densities when we consider links going from an out-group to an in-group, taking into account all such pairs. As was said, this means that the link weight distribution for such pairs should be random-like with some concentration around the mean value. That is why we define the $Kout \times Kin$ matrix of weight density P as:

$$(P)_{i,j} = p_{i,j} = \frac{\sum_{\alpha \in U_i, \beta \in V_j} w_{\alpha,\beta}}{| U_i \| V_j |}.$$

We define yet another directed random graph, corresponding to P, $\mathcal{G}P$, with independent links and with link probability: $((i, j) \in) = pui,vj)$ where ui is defined from relation: $i \in Uui$ and, similarly vj:
$j \in Vvj$. This latter graph is a kind of compressed version of the first one. $\mathcal{G}P$ should be selected in such a way that it gives a Sz. type structure in a best possible way. This is a combinatorial optimization problem with huge search space, since the number of possible partitioning is big, even in our case. evaluated by using Akaike information criteria (AIC), with minimal value giving the best choice of parameters. Thus our approach is quite similar to that in [7] where a binary directed graph was considered. The main emphasis of [7] was in predicting unknown connections based on a partially observed network. It seems that such a method can be very successful in this type of tasks. We are just interested in clusters themselves and link patterns between them. Thus the task is to structure the p2p network. However, due to computational difficulties the method is probably restricted to moderate size graps like ours. On the further hand, for a very large graph, the algorithmic version of SzRL, [10] could be cast-off, as was already done in [12] in the digital image segmentation. Such an algorithm that finds clusters indicated by SzRL runs in polynomial time with respect to the number of nodes. Probably even better algorithm would be the one sugessted in [11], that can be used even for small graphs. Unfortunately, we noticed this paper only after this paper was almost finished. However, we are looking forward applying such algorithm to find and compare Sz. clusterings in our case.

The Load-Balancing Manager

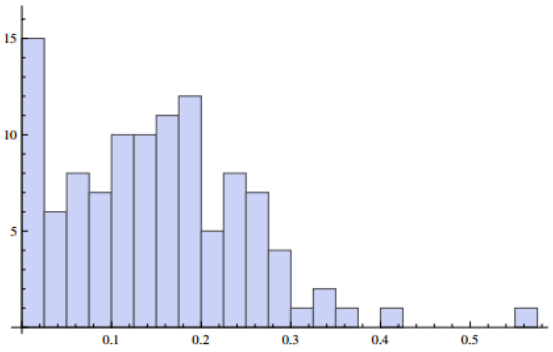
In our framework, load balancing is done both statically when a new node joins the system and an existing node leaves the system and dynamically when an existing node in the system becomes overloaded or under loaded. In static load balancing, a new node needs to find a heavily loaded node to join as an adjacent node while an existing node wishing to leave the system needs to find a lightly loaded node to shed its workload. On the other hand, dynamic load complementary is realized by either local load balancing or network load complementary. In local load balancing, an overloaded or under loaded node performs load balancing with its adjacent nodes; while in network load balancing, an overloaded or under loaded node needs to find a lightly or heavily loaded node to do load balancing. In this case, the under loaded or lightly loaded node needs to leave its current position and joins as an adjacent node of the overloaded or heavily loaded node. In specific, a node only performs network load balancing if it cannot perform local load balancing.

Histogram Structure

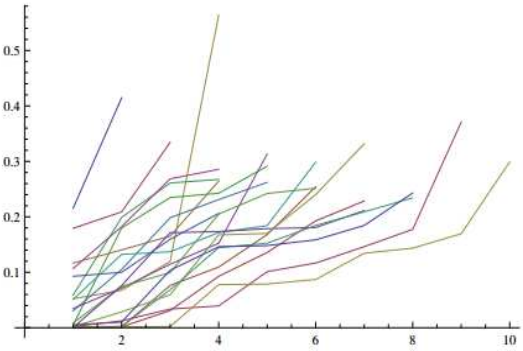
From the way a query is processed in Skip Graph, if a node x sends a query to a neighbor node y at level l, the search region is determined in one of the following ways depending on the scenarios:
1. In the first scenario, let z be a neighbor node of x at the nearest level higher than l on the same side (or direction) of x as y. Then, the search region is given by all nodes falling between y and z.
2. In the second scenario, node z as described in the first case does not exist. In this case, the search county consists of all nodes following y on the same side of x. As a result, a non-overlapping group in a

histogram in the Skip Graph can be defined as a group formed by all nodes falling between two consecutive neighbor nodes or all nodes following the farthest neighbor node of a node on the same side of that node.

Experimental Result



Distribution of weights on links for clusters. Each line represents link weights for each peer in cluster 2 pointing to cluster 1. Most lines are quite flat indicating uniformity.



Integral weights of peers within clusters. Each point gives the proportion of chunks obtained within this subgraph. High level of these magnitudes indicates that this is an important sub graph in the peer-to-peer network.Our system was quite small, yet it was large enough to prevent the use of direct exhaustive methods to find clusters. Instead we used a greedy algorithm and run it several times to find the global optimum.

Histogram Maintenance

Histogram updating in Chord is also different from other systems. When the shipment of a node changes by a factor of, the node does not send an updated information message to its neighbor nodes since neighbor nodes are not nodes having links to it. In its place, the node sends the update message to logN nodes, which are nodes preceding its 2i positions in the Chord ring (if there is a position without an available node, the adjacent node preceding that position is sent the message). If a node x receives an update request pointing at it, x checks its routing table to see if the sender is in or not. If the sender node is a neighbor of x, x recalculates its histogram and continues to send an update message to other nodes if necessary. An problem with this process is that it is expensive since a node needs approximately logN messages for every update operation.

To deal with it, we have two solutions. The first explanation is to use a variant of Chord, which chains bidirectional links such as that in [41], and hence, update messages can be sent directly to neighbor nodes and are processed as in Skip Graph and BATON. The next solution is to change the histogram update process as follows: When the load of a node changes by a factor of, the node sends updated information to only its predecessor node. When a swelling receives an update message from its immediate successor protuberance, it recalculates its histogram and continues to update its predecessor node if there is any signification change in Non-overlapping groups. Even though the second solution cannot tighten the ratio between the real average load of a group and the stored value in a histogram within with high prospect, this ratio is still in the range.

Conclusion

We proposed a framework, HiGLOB, to enable universal load balance for structured P2P systems. Each node in HiGLOB preserves the load information of nodes in the systems using histograms. This allows the system to have a global view of the load distribution and hence facilitates global load balancing. We partition the system into non-overlapping groups of nodes and maintain the average load of them in the histogram on a node. We also proposed two techniques to reduce the overhead of maintaining and assembling histograms. Even though the proposal is a general framework, it remains possible to deploy different kinds of P2P systems on it. We demonstrated this by structure three well-known structured P2P systems: Skip Graph, BATON, and Chord on our proposal. Our routine evaluation shows that our HiGLOB enabled systems are superior over other methods.

References

[1] Madhukar and C. Williamson, “A Longitudinal Training of P2P Traffic Organization,” Proc. Int’l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm.Systems (MASCOTS), 2006.

[2] D. Karger, F. Kaashoek, I. Stoica, ..R. Morris, and H. Balakrishnan, “Chord: A Scalable Peer-to-Peer Lookup Service for

Internet Applications," Proc. SIGCOMM '01, pp. 149-160, 2001.

[3] A. Rao, K. Lakshmi narayanan, S. Surana, R. Karp, & I. Stoica, "Load Balancing in Structured P2P Systems," Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS), 2003.

[4] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," Proc. ACM Symp. Parallelism in Algorithms and Architectures (SPAA), 2004.

[5] P. Ganesan, M. Bawa, and H. Garcia-Molina, "Online Balancing of Range-Partitioned Data with Applications to Peer-to-Peer Systems," Proc. Very Large Databases Conf. (VLDB '04), pp. 444-455, 2004.

[6] J. Aspens and G. Shah, "Skip Graphs," Proc. 14th Ann. ACM-SIAM Symp. Discrete Algorithms (SODA '03), pp. 384-393, 2003.

[7] D. Karger and M. Ruhl, "Simple Efficient Load Harmonizing Algorithms for Peer-to-Peer Systems," Proc. Int'l Workshop Peer to- Peer Systems (IPTPS), 2004.