An Efficient Client Server Assignment for Internet Distributed Systems

Swathi Balakrishna, Dr. Ling Ding
Computer Science and Systems,
University of Washington, Tacoma

***Abstract*—Internet is a network of several distributed systems that consists of clients and servers communicating with each other directly or indirectly. In order to improve the performance of such a system, client-server assignment plays an important role. Achieving optimal client-server assignment in internet distributed systems is a great challenge. It is dependent on various factors and can be achieved by various means. Our approach is mainly dependent on two important factors, 1) Total communication load and 2) Load balancing of the servers. In our approach we propose an algorithm that is based on Semidefinite programming, to obtain an approximately optimal solution for the client-server assignment problem. Our simulation results indicate that the proposed algorithm outperforms other client server assignment algorithms in terms of optimal value and running time.**

***Index Terms*—Communication Load, Distributed Systems, Load balancing, Semidefinite programming.**

I. Introduction

THE modern internet is a collection of interconnected networks of various systems that share resources. An ideal distributed system provides every node with equal responsibility, and nodes are similar in terms of resource and computational power. However in real world scenarios it is difficult to achieve it as such systems includes overhead of coordinating nodes, resulting in lower performance. Typical distributed system consists of servers and clients and servers are more computational and resource powerful than clients. Typical examples of such systems are e-mail, Instant messaging, e-commerce, etc. Communications between two nodes happen through intermediate servers. When node A sends mail to another node B, communication first flows from A to its email server. Email server is responsible for receiving and sending emails, from and for the clients assigned to it. Node A's email server sends data to node B's email server, which in turn is responsible for sending mail to node B. Email servers communicate with each other on behalf of their client nodes. Clients are assigned to a server based on various parameters like organizations, domains, etc. In our paper we solve client-server assignment problem based on total communication load and load balancing on servers.

Client server assignment can be designed based on the following observations;

1. If two clients are assigned to the same server, the server will receive message from one client and forward to another one. If they are on different servers, the sender client first sends to its server. The sender's server will receive data and forward it to the receiver's server. The receiver server will receive data and forward it to receiver client. Thus the total communication between two frequently interacting clients increases if they are assigned to two different servers. Assigning these two clients to a single server makes all information exchange locally. It is more efficient to have all the clients assigned to few servers to minimize total communication.
2. On the contrary having fewer servers, results in those servers being heavily loaded while others are left underutilized. If a server is heavily loaded it results in low performance due to excessive resource usage on that server. Thus it is necessary to consider load balance on the server while assigning clients.

From the above observations it is clear that total communication load and load balancing are two contradicting features. Hence equilibrium must be maintained between overall communication load and load balancing of the servers. This paper solves the client-server assignment problem based on Semidefinite programming. Communication pattern among the clients is taken as input and it outputs an approximately optimal client server assignment for a pre-specified tradeoff between load balancing and total communication.

Client Server assignment problem is relevant in current world for a host of applications from e-commerce, social networking to online auctioning. Group of friends tend to exchange lot of information, stories and photos among themselves rather than with strangers. Thus, assigning group of friends to a single server reduces the inter-server communication; meanwhile it is important to balance the communication load. This is a classic example of optimal client server assignment problem. In distributed database systems, assigning the search keywords that are frequently queried together to the same server improves the performance by reducing the inter-server communication.

Semidefinite programming is a sub field of convex optimization. Semidefinite programming has recently emerged to prominence because it admits a new class of problem previously unsolvable by convex optimization techniques, [7] and it theoretically subsumes other convex techniques.

II. RELATED WORK

Client Server assignment problem can be approached based upon various factors. There has been lot of research in this area.

1. Clustering Algorithms

Client Server assignment problem is an instance of clustering problem. Clients and their communication pattern can be denoted as a graph, with vertices representing the clients and the edges between two vertices representing the communication between the respective clients. Communication frequency between two clients can be represented by the weight of the edge between the corresponding vertices. Clustering algorithm forms fixed number of clusters of clients based on a given objective. Objective of the clustering algorithm in this paper is to minimize the amount of inter group communication and balance the sum of weights of all edges in the group.

Normalized cuts (NC) [1] is a clustering algorithm that partitions an undirected graph into two disjoint partitions such that

$$F_{ncut} = \frac{W_{1,2}}{W_{1,1}+W_{1,2}} + \frac{W_{2,1}}{W_{2,2}+W_{2,1}} \quad \text{is minimized,}$$

Where $W_{i,j}$ is the sum of weights of all edges that connects the vertices in group i and j.

In order to solve Fncut efficiently, NC uses Eigen values of adjacency matrix. But NC tends to cause imbalance in the volumes of the groups by isolating the vertices that do not have strong connection to others.

Balanced clustering algorithms for power law graphs are examined in [2]. Author concludes that combining multiple trials of a randomized flow-based rounding methods and solving a semi definite program yields effective results.

In [3] it is shown that the optimal client server assignment for pre-specified requirements on total communication load and load balancing is NP-hard. A heuristic algorithm based on relaxed convex optimization is used for finding the approximate solution to the client server assignment problem

2. Distributed Interactive applications (DIAs)

DIAs are networked systems that allow multiple participants at different locations to interact with each other. Wide spread of client locations in large-scale DIAs often requires geographical distribution of servers to meet the latency requirements of the applications. In the distributed server architecture, client assignment to servers directly affects the network latency involved in the interactions between clients. [4] focuses on client assignment problem for enhancing the interactivity performance of DIAs. In this paper, the problem is formulated as a combinational optimization problem on graphs and proved to be NP-complete of assigning clients to appropriate servers. Three heuristic algorithms are proposed, namely, nearest assignment, Greedy assignment, Distributed greedy assignment and are compared and evaluated using real Internet latency data.

[5] is an improved version of [4] and the results show that the proposed algorithms, nearest Assignment algorithm, Modify Assignment and Distributed modify-assignment, are efficient and effective in reducing the interaction time between clients. However, both [4] and [5] consider the client assignment in DIAs and do not consider the load on the server and other factors.

3. Load-distance balancing Problem(LDB)

Client server assignment in [6] is modeled on the fact that the delay incurred by a client is dependent on load of the server and the distance to the assigned server. Delay on the client side is the sum of network delay (proportional to distance to its server) and congestion delay at the server. Hence it focuses on distance between the client and the server and the load on the server in client-server assignment. The problem is defined as Load-distance balancing (LDB) problem.

This paper targets 2 flavors of LDB. In the first flavor, the objective is to minimize the maximum incurred delay using an approximation algorithm and an optimal algorithm. In the second flavor, the objective is to minimize the average incurred delay, which the paper mentions it as NP-hard and provides a 2-approximation algorithm and exact algorithm.

However to the best of our knowledge there is no clustering algorithm that is designed as a Semidefinite programming problem to achieve our goal: Load balancing on servers and minimizing the communication load between servers, for client-server assignment.

III. PRELIMINARY

In this paper, we have used Semidefinite programming as the core algorithm to solve the problem. It is an optimization problem with a linear objective function over a spectrahedron and supports many properties like linear programming. To solve a problem using Semidefinite programming, Feige et. al divided the procedure into several steps in [8] as follows:

1) Formulate the problem satisfying integer quadratic programming.

2) Relax the integer variables x1… xn to real variables y1….yn. Change the formula to Vector Programming with variables v1….vn, and change the vector programming satisfying Semidefinite programming.
3) Solve the relaxation Semidefinite programming in polynomial time optimally. Obtain the vectors v1…vn through Cholesky Factorization.
4) Use rounding method to get the solution to integer quadratic programming. The rounding methods can be threshold rounding, randomized rounding, etc.

IV. Model of Communication

To solve this problem of client-server assignment we first design a model for communication. Every client is assigned to a single server. When a client has to interact with another client it passes on message to its assigned server and server forwards that message. If the destination client is associated with same server as source client, message is forwarded directly from the server to destination client. If the destination client is assigned to another server, message is forwarded to destination client server from source client's server.

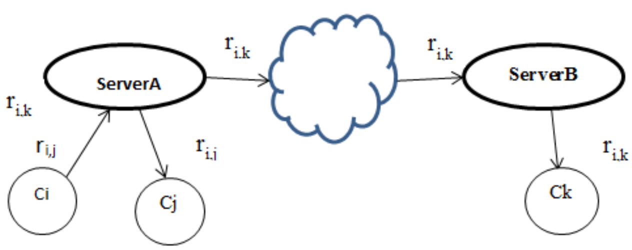

Fig 4-1

In fig 4-1 we have 2 servers, Servers A and B and 3 clients Ci, Cj and Ck. Ci and Cj are connected to the same server, Server A. When client Ci wants to communicate with client Cj, Ci forwards the message to its associated server A. Server A forwards the message to client Cj. In process of message forwarding any server performs only two actions, reception and forwarding. If r is the size of message, then load on the server is increased by r for reception and forwarding. As Ci and Cj are assigned to same server, communication between them increases the load on Server A by r. Consider the communication between clients Ci and Ck. Load on Server A increases by r as Server A receives message from Ci and forwards to Ck's associated server. Load on Server B is also increased by r as Server B also receives and forwards the message to Ck. Therefore overall load increases by 2r when clients are assigned to two different servers as compared to r in case of single server.

Based on this communication model we solve the client assignment problem. Initially we develop the algorithm for client assignment in distributed system with only 2 servers. Then we will expand the algorithm for n servers.

V. Algorithm for 2 Server

In this paper we focus on total communication load and load balancing in client server assignment. Our goal is to achieve client assignment minimizing the overall load and load balance of servers. In this section we consider client assignment in a system with 2 servers, Server A (S_A) and Server B (S_B). Load issue is solved using Semidefinite programming. This solution is extended to case of multiple servers.

X_i variable is used to denote the server to which client i is assigned to. Formula 1 defines Xi,

$$x_i = \begin{cases} 1 & \text{client } i \text{ is assigned to } S_A \\ -1 & \text{otherwise} \end{cases} \quad (1)$$

Let $r_{i,j}$ denote the date rate of communication from client i to client j. Communication load will be $r_{i,j}$ between clients i and j if they are assigned to the same server. Communication load will be doubled if clients i and j are assigned to two different servers. Formula for total communication load C_{total} is as given below.

$$C_{total} = \sum_i \sum_j r_{i,j} + \sum_i \sum_j \frac{1 - x_i x_j}{2} * r_{i,j} \quad (2)$$

Note that $r_{i,i} = 0$.

Load balance of the servers is the load difference between the two servers S_A and S_B. $C_{balance}$ is use to denote the load balance of servers. If two communicating clients are assigned to two different servers, they increase the communication load on both servers equally resulting in no impact on load balance.

In figure 4-1, clients C_i and C_k are assigned to two different servers. When C_i has to communicate with C_k, C_i sends message to S_A which is forwarded to S_B. S_B receives from S_A and forwards it to C_k. This increases the communication load on S_A and S_B but load difference between S_A and S_B does not change. From this observation it is clear that to calculate the load balance of a server we need to calculate only cost of communications between those clients which are assigned to that server. We denote load on S_A as C_A and load on S_B as C_B. C_A and C_B includes load of communications between clients assigned to only S_A and S_B respectively.

We can express load on each server S_A and S_B based on formula 1 as follows:

$$C_A = \sum_{i,j} \frac{1 + x_i}{4} * r_{i,j} + \sum_{i,j} \frac{1 + x_j}{4} * r_{i,j} \quad (3)$$

$$C_B = \sum_{i,j} \frac{1 - x_i}{4} * r_{i,j} + \sum_{i,j} \frac{1 - x_j}{4} * r_{i,j} \quad (4)$$

Hence the load difference between Server A and Server B can be expressed as $D = |C_A - C_B|$.

D can also be expressed as,

$$D = \left| \sum_{i,j} \frac{x_i + x_j}{2} * r_{i,j} \right| \quad (5)$$

D, load balance cannot be added directly to our objective function as this makes our objective function non-quadratic. Non-quadratic functions are hard to solve hence we modify our load balance. New load balance is denoted by $|C_A - C_B|^2$ or$C_{balance} = (C_A - C_B)^2$. Minimizing $C_{balance}$ results in a minimized value of D, load balance.

To explain our objective function better, we express D in other two equivalent formulas:

$$D = \left| \sum_{i} x_i * s_i \right| \quad (6)$$

$$s_i = \sum_{i} \frac{1}{2} * (r_{i,j} + r_{j,i}) \quad (7)$$

We obtain,

$$C_{balance} = \sum_{i} s_i^2 + \sum_{i \neq j} (x_i x_j) * (s_i s_j) \quad (8)$$

Since $x_i * x_i = 1$.

Our goal is to minimize both total load and the load balance among the servers. Objective function is expressed as,

$$\begin{aligned} &minimize: \propto C_{total} + (1-\propto) C_{balance} \\ &subject\ to: x_i^2 = 1, x_i \in Z. \quad 1 \leq i \leq n \end{aligned} \quad (9)$$

Our objective function of (9) is a strict quadratic formula along with the constraints, as it is clear from formulas (2) and (8) that C_{total} and $C_{balance}$ consists of monomials of either degree 0 (constants) or degree 2.

We define two new parameters, $w_{i,j}$ and C, which are obtained by denoting C_{total} and $C_{balance}$ by (2) and (8) in the objective function of (9).

$$w_{i,j} = -\frac{1}{2} \propto r_{i,j} + (1-\propto) s_i s_j \quad (10)$$

$$C = \frac{3}{2} \propto \sum_{i \neq j} r_{i,j} + (1-\propto) \sum_{i} s_i^2 \quad (11)$$

Objective function of (9) can be expressed as:

$$minimize: \sum_{i \neq j} w_{i,j} x_i x_j + C \quad (12)$$

We translate our original problem to a Vector problem. To solve (12) based on constraints of (9), we relax the n integer variables to n vector variables in R^n and substitute each term with degree 2 to the inner product of two vector variables. According to [8] a vector program is a program with all the variables as vectors within R^n, say $v_1 \ldots v_n$, and objective functions and all constraints are linear functions of inner products $v_i . v_j$, $1 \leq i \leq n$. Thus we have our original problem transferred into the following vector program:

$$minimize: \sum_{i \neq j} w_{i,j} (v_i .\ v_j) + C$$

$$\begin{aligned} subject\ to: \quad & v_i . v_i = 1, && v_i \in V \\ & v_i \in R^n, && v_i \in V \end{aligned} \quad (13)$$

For any feasible solution of (9) if we assign a vector form feasible solution for (13) is obtained. Vector programming is equivalent to Semidefinite programming. For a matrix $A \in R^{nXn}$, let $a_{i,j}$ denote the (i,j)th entry in matrix. Trace of A, denoted by tr(A) is defined as the sum of its diagonal entries. Given two matrices $A, B \in R^{nXn}$, trace of $A^T B$ is given by, $tr(A^T B) = \sum_i \sum_j a_{i,j} b_{i,j}$. Let U be a symmetric and positive Semidefinite matrix and W be a matrix with all diagonal entries as 0 and other entries as $w_{i,j}$ defined in (10). We obtain the equivalent Semidefinite programming as follows:

$$\begin{aligned} minimize: \quad & tr(W^T U) \\ subject\ to: \quad & u_{i,i} = 1, \quad 1 \leq i \leq n, \\ & \text{U is positive Semidefinite} \end{aligned} \quad (14)$$

By solving (14) we obtain a solution matrix U_{sol}. Then by using Cholesky Factorization we compute a matrix V = $(v_1, v_2, \ldots v_n)$ that satisfies $V^T V = U_{sol}$ in $O(n^3)$ time. V is set of all vectors in solution of relaxed vector programing in (13). To obtain the final solution of our original problem, we have to round the n vectors $(v_1 v_2 \ldots v_n)$ to $(x_1 x_2 \ldots x_n)$. This rounding method is similar to the procedure of solving MAX-CUT problem: randomly chose a uniformly distributed vector r in n-dimensional unit sphere, then for $v_i.r \geq 0$, then x_i, the rounded result of v_i, is set to 1, else x_i is set to -1.

In this section we have solved the case of 2 server scenario. We extend this algorithm for n-server scenario.

VI. SOLUTION FOR N-SERVER SCENARIO

In a real world scenario we have several servers. Here we extend our 2-server algorithm for N servers. Basis of our N server algorithm is first split the group of servers into 2 groups. We treat each group as a server and apply our 2 server algorithm to assign clients to either one of the group in a balanced way. We further split the group with more than one server to two groups and apply the 2 server algorithm. This recursive process is continued until every group contains only one server with clients assigned to it. This way we solve the general case problem.

Biggest challenge in solving this problem is to split the servers. Given 'K' servers ideal way to solve is to split them into 2 groups with equal number of servers. Load is balanced among the two groups with our algorithm and as the group is evenly distributed it ensures further assignment of clients to small groups evenly. But this ideal situation of equal number servers in the group is achieved only when 'K' is power of 2.

If 'K' is even we can split the servers into two groups with equal size of servers as $\frac{K}{2}$ in each group. When 'K' is odd we try to split the servers as evenly as possible, with size of one group as $\left\lfloor\frac{K}{2}\right\rfloor$ and other with size as $\left(K-\left\lfloor\frac{K}{2}\right\rfloor\right)$ which is $\left\lceil\frac{K}{2}\right\rceil$. It is clear that as we proceed we see lots of groups with unequal sizes. We have to reconsider how to achieve load balance between server groups. Consider two server groups g_1 and g_2, and let $|g_1|$and $|g_2|$ be the number of servers respectively. If load assigned to both the groups are e, then theoretically each server in g_1 gets load of $\frac{e}{|g_1|}$ and each server in $g_2$gets load of $\frac{e}{|g_2|}$. As $|g_1| \neq |g_2|$ and if $|g_1| < |g_2|$ then each server in group g_1 gets heavier load than servers in g_2 or vice versa. Hence we see that neither of them is well load balanced. As goal of our problem is to balance the load of each server we have to assign groups of unequal size with different loads.

Consider a system with K server, split into two groups, group A with K_A servers and group B with K_B servers $(K_A + K_B = K)$. Let C_A and C_B represent the loads on the group, then load difference between these two server groups is:

$$D = \frac{2}{K}|K_B C_B - (K - K_B)C_A|$$

Using (3) and (4), we get,

$$D = \left|\frac{2}{K}\left\{\frac{K}{2}\sum_{i,j}\frac{x_i + x_j}{2} * r_{i,j} - \frac{K_A - K_B}{2}\sum_{i,j} r_{i,j}\right\}\right| \qquad (15)$$

If K is even we have $D = \left|\sum_{i,j}\frac{x_i+x_j}{2} * r_{i,j}\right|$ which is same as (5) as $K_A = K_B = {}^{K}/_{2}$.

In order to minimize D, we have to make $\frac{K}{2}\sum_{i,j}\frac{x_i+x_j}{2} * r_{i,j}$ as close to $\frac{K_A-K_B}{2}$ as possible i.e. approach $\sum_{i,j}\frac{x_i+x_j}{2}$ to $\frac{K_A-K_B}{2}$.

From the above analysis if K is odd, without loss of generality $K_A = \left\lceil\frac{K}{2}\right\rceil$ and $K_B = \left\lfloor\frac{K}{2}\right\rfloor$, $K_A - K_B = 1$, $C_{balance}$ approach to $\left(\frac{1}{K}\right)^2$. Thus problem can be formulated as,

$$minimize:\ \propto C_{total} + (1-\propto)\left|C_{balance} - \frac{1}{K^2}\right|$$

$$subject\ to: x_i^2 = 1, x_i \in Z. 1 \le i \le n \qquad (16)$$

We relax each integer variable x_i to a vector variable v_i like in section (V). Let C_{total}^v and $C_{balance}^v$ represent the expressions after the relaxation of (2) and (8) respectively. We have an absolute value item in the objective function of relaxed Vector programming. General square of the item is calculated to get rid of calculation of absolute value, but it will make the problem unsolvable by Semidefinite Programing. Since C_{total}^v is always greater than 0, here we split the relaxed vector programming problem into two sub-problems:

$$\begin{aligned} &minimize:\ \propto C_{total}^v + (1-\propto)\left(C_{balance}^v - \tfrac{1}{K^2}\right) \\ &subject\ to:\ v_i^2 = 1, 1 \le i \le n; \\ &\qquad C_{balance}^v \ge {}^{1}/_{K^2} \end{aligned} \qquad (17)$$

$$\begin{aligned} &minimize:\ \propto C_{total}^v + (1-\propto)\left(\frac{1}{K^2} - C_{balance}^v\right) \\ &subject\ to:\ v_i^2 = 1, 1 \le i \le n; \\ &\qquad C_{balance}^v \le {}^{1}/_{K^2} \end{aligned} \qquad (18)$$

We define $\sim w_{i,j} = (\propto -1)s_i s_j - \frac{\propto}{2} r_{i,j}$ as done in section (V). Let Z be a $n \times n$ matrix with (i,j) th entry as $s_i s_j$. Let W_1 be same matrix as W and let W_2 be a $n \times n$ matrix with (i,j) th entry as $\sim w_{i,j}$. We obtain the corresponding 2 Semidefinite Programming problem as:

$$\begin{aligned} &minimize: && tr(W_1^T U) \\ &subject\ to: && u_{ii} = 1, 1 \le i \le n; \\ & && tr(Z^T U) \le \tfrac{1}{K^2} - \textstyle\sum_i s_i^2 \\ & && U\ is\ semidefinite \end{aligned} \qquad (19)$$

$$\begin{aligned} &minimize: && tr(W_2^T U) \\ &subject\ to: && u_{ii} = 1, 1 \le i \le n; \\ & && tr(Z^T U) \ge \tfrac{1}{K^2} - \textstyle\sum_i s_i^2 \\ & && U\ is\ semidefinite \end{aligned} \qquad (20)$$

We compare the two objective function values obtained by solving the above two sub problems. The one with the smaller value is chosen and the corresponding $(v_1, \ldots v_n)$ is the solution to the relaxed vector programming function. Then we obtain the final solution $(x_1, \ldots x_n)$ of (16).

VII. SIMULATION RESULTS

We compare the results of simulation of our algorithm with the simulation results of algorithm presented in [3]. In [3] as convex optimization is used we abbreviate algorithm in our table and graphs as CO. To display our results we abbreviate our algorithms as SDP. Fig 7-1 displays the client assignment of ten clients to two servers in a graphical form using both the algorithms with α =0.5.

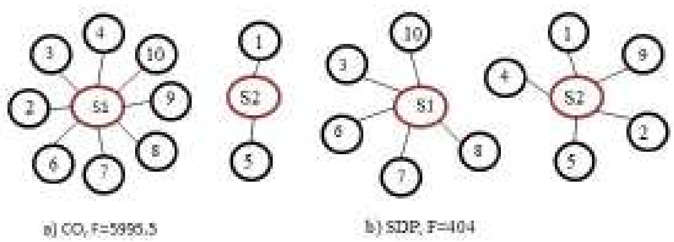

Fig. 7-1

Table 7-1 gives the comparison of values obtained for F, Fc and Fl. It is evident that the optimal value F obtained through our algorithm is smaller when compared with [3] with a smaller runtime.

N= 10, α = 0.5	CO	SDP
F	5995.5	404
Fc	110.0	788
Fl	11880	20
Runtime	0.358	0.249

Table 7-1

We conducted the experiments for various values of N (number of clients). With greater N it was clearly evident that the run time to solve our algorithm was much smaller when compared to algorithm presented in [3]. Table 7-2 to 7-6 gives comparison of results obtained for various N values.

N = 20, α = 0.5	CO	SDP
F	14365	1740
Communication Load	170	3291
Load Balance	28560	189
Runtime	0.9516	0.8424

Table 7-2

N = 50, α = 0.5	CO	SDP
F	148240	10256
Communication Load	545	20024
Load Balance	295935	488
Runtime	2.7640	2.4804

Table 7-3

N = 100, α = 0.5	CO	SDP
F	543402	41779
Communication Load	1043.00	81447
Load Balance	1085761.8	2111
Runtime	4.045	3.0436

Table 7-4

Table 7-5 and Table 7-6 contains optimal obtained for same rate matrix for 200 clients but with different alpha value.

N = 200, α = 0.5	CO	SDP
F	2602610	175529
Communication Load	2282.0	327446
Load Balance	5202944.1	23612
Runtime	16.224	13.75

Table 7-5

N = 200, α = 0.7	CO	SDP
F	1562480	167688
Communication Load	2282.0	328738
Load Balance	5202940.4	6638
Runtime	15.91	12.15

Table 7-6

Our algorithm has a better run time and F value when compared with the algorithm used in [3]. As we apply our algorithm for bigger values of N we can observe that the problem is solved efficiently with a smaller runtime.

VIII. CONCLUSION

In this paper we present an approximation algorithm for solving the client server assignment problem in internet distributed systems. Our simulation results indicate that the proposed algorithm outperforms the optimal value and running time obtained by client server assignment algorithm presented in [3].

REFERENCES

[1] J. Shi and J. Malik, “Normalized cuts and image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22,no. 8, pp. 888–905, 2000.

[2] “Finding good nearly balanced cuts in power law graphs,” Yahoo Research Labs, Tech. Rep., 2004.

[3] Nishida, H.; Thinh Nguyen; , "Optimal Client-Server Assignment for Internet Distributed Systems," Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on , vol., no., pp.1-6, July 31 2011-Aug. 4 2011

[4] Lu Zhang; Xueyan Tang; , "Client assignment for improving interactivity in distributed interactive applications," INFOCOM, 2011 Proceedings IEEE , vol., no., pp.3227-3235, 10-15 April 2011

doi: 10.1109/INFCOM.2011.5935173

[5] Zhang, L.; Tang, X.; , "Optimizing Client Assignment for Enhancing Interactivity in Distributed Interactive Applications," Networking, IEEE/ACM Transactions on , vol.PP, no.99, pp.1, 0 doi: 10.1109/TNET.2012.2187674

[6] Bortnikov, E., Khuller, S., Li, J., Mansour, Y. and Naor, J. S. (2012), The load-distance balancing problem. Networks, 59: 22–29. doi: 10.1002/net.20477

[7] Stephen Boyd, Laurent El Ghaoui, Eric Feron, and Venkataramanan Balakrishnan. Linear Matrix Inequalities in System and Control Theory. SIAM, 1994.

[8] U. Feige and M. Langberg. The rpr2 rounding technique for Semidefinite programs. J. Algorithms, 60(1):1–23, 2006.