Load Balancing in Structured P2P Systems using Server Reassignment Technique

S.S.Patil
Rajarambapu Institute of Technology Rajaramnagar/CSE
Sangli, India

S.K.Shirgave
Dattajirao Kadam College of Textile Engineering Inchalkaranji/CSE
Sangli, India

ABSTRACT
Load balancing among application layer peer-to-peer (P2P) networks is critical for its effectiveness but, is considered to be the most important development for next-generation internet infrastructure. Most structured P2P systems rely on ID-space partitioning schemes to solve the load imbalance problem and have been known to result in an imbalance factor of θ (log N) in the zone sizes.

Two important contributions to minimize the same are proposed in [1]. First, the virtual-server-based load balancing problem using an optimization-based approach and deriving proposal in general and its advantages over previous strategies. Second, characterizing the effect of heterogeneity on load balancing algorithm performance and the conditions in which heterogeneity may be easy or hard to deal with based on an extensive study of a wide spectrum of load and capacity scenarios.

Keywords: Distributed Hash Table, Load Balance, Local Search, Structured Peer-To-Peer System, Generalized Assignment Problem.

I. INTRODUCTION

P2P system harnesses the resources of large populations - networked computers in a cost-effective manner such as the storage, bandwidth, and computing power. In structured P2P systems, data items are spread across distributed computers (nodes), and the location of each item is determined in a decentralized manner using a distributed hash lookup table (DHT) [2], but there are several problems in it. Following [4], let N be the number of nodes in the system and fmax be the ratio of the largest zone size to the average zone size. It is known that fmax is θ (log N) with high probability [4] and this could result in a θ (log N) load imbalance factor in the number of objects even when nodes have homogeneous capacities. But in practice, the resources of P2P systems are most likely overlaid on top of peer nodes with extreme heterogeneity in hardware and software capabilities. So they cannot adapt to dynamic workload changes in real networking conditions. To resolve the above problems, approach on the virtual server (VS) i.e. called as a migration-based approach to load balancing is focused in [1], which helps to under load the overloaded physical node by moving portions of the load dynamically. A VS acts as a peer in the original DHT architecture which is responsible for a zone, but each physical node may be associated with several VS's.

Some noteworthy points in the VS concept are:

- The transfer of VS from one physical node to another is equivalent to a leave - followed by a join operation to the underlying DHT which is supported by the DHT framework.
- The concept is applicable to many types of resources such as storage, CPU processing time, bandwidth, etc.

The major contribution in [1] is to derive an effective server reassignment algorithm for the solution of the load balance problem and address issues as:

- Node registration policies.
- Effect of the number of directories - vital to success of the framework.

Other major contribution is to systematically investigate the effect of forms of heterogeneity on the difficulty of the server reassignment problem.

II. OVERVIEW OF RELATED WORK

Recently there is dynamic interest for "The load balance problem for heterogeneous overlay" in the research community. In [1] focus on the notion of VS's is mostly based on [3], whose explicit definition and use for load balance were first proposed by Rao et al. [5]. In [5] author introduce several load balancing algorithms, including many-to-many with dislodge (called M2M in this paper) and M2M without dislodge (called DM2M), based on the assistance of a new type of peer node, called directory. M2M is shown to be superior to DM2M in performance.

Both algorithms are intuitively appealing, but there are several important issues with the general M2M approach:

- The algorithms are unable to find feasible assignments in certain very simple situations, such as starting from a set of VSs S, the M2M strategy searches only in the direction of decreasing total load in S. Presumably, this strategy guarantees algorithm termination.
- A number of design issues, such as how nodes should register with directory nodes, are not adequately addressed.

Fisher et al. [6] proved that the generalized assignment problem is NP complete. Lourenco and Serra proposed a general framework whose strategy is adopted in [1] algorithms, albeit with much modification due to the much larger search space for problem.

The formulation of the integer programming GAP is shown as follows:

$$Minimize\ \mathrm{f(x)} = \sum_{i \in I} \sum_{j \in J} \left(c_{ij}\, x_{ij}\right) \quad (1)$$

s.t.

$$\sum_{i \in I} l_i\, x_{ij} \leq t_j \ \forall \mathrm{j} \epsilon \mathrm{J} \qquad (2)$$

$$\sum_{j \in J} x_{ij} = 1, \forall \mathrm{i} \epsilon \mathrm{I} \qquad (3)$$

$$x_{ij} = \begin{cases} 1, & \text{if task i is assigned to agent j,} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

Figure 1. Gap formulation

The objective of GAP is to find a solution that minimizes the total cost, as defined in (1). Equation (2) enforces the resource capacity constraints of the agents. Equation (3) guarantees that each task is assigned to one and only one agent.

The general strategy in the GAP proposals is the creation of a solution and local searching in the "vicinity" of the created solution before moving on to the next iteration. The notion of neighborhoods and moves for conducting a local search with respect to a created solution are needed. A shift move consists of removing a task (i.e. VS) from one agent (node) and assigning it to another, whereas a shift neighborhood is a set of such moves.

- Simple shift neighborhood and a restricted ejection chain of length 2 neighborhood were first employed by Lourenco and Serra [7]. Both neighborhoods are searched using a single cost function, making no distinction between feasible and infeasible solutions.
- The fundamental insight in [1] is on need of more focused definition of the neighborhood search spaces for the local search.
- The authors in [1] proposes to classify the search spaces into two types using dual-pace local search algorithm and develop a corresponding search strategy as an effective approach to tackle the problem.

III. PROPOSED ALGORITHM

The overall algorithm proposed in [1] is called dual space local search (DSLS) and is based on the framework by Lourenco and Serra [7], is discussed below:

- Construction The algorithm invokes the ant system heuristic (ASH) algorithm to construct an initial solution x for the current iteration.
- Improvement The algorithm then invokes the DLS procedure to derive a local minimum solution based on the initial solution. DLS is another iterative loop comprising two main phases for searching in the two ejections-chain neighborhoods:
  a) First, if the initial solution generated in the first step is not a feasible solution, the algorithm performs a local search procedure in a feasibility-improving ejection-chain neighborhood N(x) to adjust the solution to a feasible one x'.
  b) Second, based on the feasible x', the algorithm performs another local search in a cost-reducing ejection-chain neighborhood N'(x) to adjust x' to a lower cost one x''.
- Pheromone trail update The current best solution will be replaced by x if it is the one. The pheromone trails will be updated to reflect the effect of x''.

The rationale for the design is that, even though ASH is an effective randomized restart procedure that can construct a good initial solution in reinforcement style of learning using the pheromone trails, it is not as effective for finding nearby local optima solutions several steps away from the generated solution. A local search algorithm must be used to improve the constructed solution to enhance the search in terms of earlier detection of high-quality solutions.

DLS algorithm is the local search component that finds the local optima solution in the neighborhood of a given initial solution. In addition, distinguish between the search spaces by the purpose they are to achieve and thus can search in significantly smaller search spaces.

A. The Ant System Heuristic :

The algorithm is as follows:

The tasks are assigned to the agents in a greedy way, where the assignment is done biased by the tij . A task i is assigned to a particular agent j in the following way:

1. With probability p_0 , choose the agent j* with maximal value of t_{ij} .
2. With probability 1- p_0 , choose the agent j* according to the following probability distribution:

$$p_{ij} = \frac{ij}{\sum_{l \epsilon Li}\, il} \text{ if j } \epsilon\ L_i, \text{or 0 otherwise}$$

This assignment constitutes the first step in the general framework, followed by a local search that tries to improve this initial solution.

3. The pheromone trails are updated in the following way:

$$\mathrm{t_{ij}^{new}} = \mathrm{t_{ij}^{old}} + \Delta_{\mathrm{ij}}\text{, where r , } 0 < \mathrm{r} < 1,$$

is the persistence of the trail, i.e. 1 - r , represents the evaporation.

Also, the updated amount is:

$$\Delta_{ij} = \begin{cases} \max * \mathrm{Q}, & \text{if task i is assign to agent j in the solution,} \\ 0, & \text{OTHERWISE} \end{cases}$$

$$Where\ Q = \begin{cases} 0.01, & \text{if the solution is infeasible,} \\ 0.05, & \text{if the solution is feasible} \end{cases}$$

4. Note that, if the solution is feasible, the pheromone trail has a bigger increment, trying to give greater probability to feasible assignments.

B. Two-Stage Descent Local Search Procedure :

DLS is itself an iterative loop whose body consists of two main phases for searching in two ejection-chain neighborhood search

spaces. First, a feasibility-improving neighborhood N(x) is generated to search for a feasible solution in the vicinity of x. Second, a cost-reducing neighborhood N'(x) is generated to find lower cost solutions in the vicinity of a feasible solution. Next, the two ejectionchain neighborhoods and their search strategies are presented.

C. Dual Neighborhood Search Spaces :

The ejection chain neighborhood can be obtained by the application of the following two types of moves:

Move A: Remove a task i from an agent (agent j), then insert this task i in a different agent (agent w).

Move B: Remove a task i from an agent (agent j), then insert this task i in a different agent (agent w). After, remove a task k from agent w (w different from j) and insert task k in another agent (different from w, but it can equal of j).

The ejection chain neighborhood of a solution can be obtained in a similar way as it was done for the shift neighborhood, but move of type B is only applied if the move of type A was not successful. Note also that the number of neighbors is of order $O(n^2m^2)$, a larger number than the shift neighborhood.

The simple overflow function f'(x), is used to define the search spaces. The definitions of i, j, I, J, tj, lij, and xij are the same as those in (2), (3), (4), and (5). This function measures the extent of capacity violation and is zero for a solution x if it is feasible:

$$f'(x) = \sum_{j \epsilon J} \max\left(0, \sum_{i \epsilon I} l_{ij}\, x_{ij} - t_j\right)$$

If an initial solution x is not feasible, that is, f'(x) > 0, the feasibility-improving ejection-chain neighborhood N(x) is defined to be the tuples or moves (i, j, i'; j'; j''):

$$N(x) = \{(i, j, i', j', j'') \mid x_{ij} = 1, x_{i'j'} = 1, s_j > t_j, s_{j'} < t_{j'}, i \neq i', j \neq j'\}$$

where $s_j = \sum_{i'' \epsilon I} x_{i''j}$, $s_{j'} = \sum_{i'' \epsilon I} x_{i''j'}$, $i, i' \epsilon I$, and $j, j', j'' \epsilon J$

The main steps of the descent local search are:

1. Obtain an initial solution x (using the ASH).
2. Let flag=false;
3. Neighborhood(x);
4. If flag=false, stop (a local optimum was found), otherwise repeat step 3.

D. Update the Current Best Solution and Pheromone Trail Variables :

The current best solution x* will be replaced by the solution x produced in the iteration, if it is found to be better. A solution is considered to be better if it satisfies one of the following two conditions:

- f'(x) = 0, f'(x*) = 0, and f(x) < f(x*): Both the new solution x and the current best solution x* are feasible (f'(x) = 0; f'(x*) = 0) and the cost of the new solution is lower than that of the current best solution.
- f'(x) < f'(x*): The overflow amount in the new solution is smaller than that of current best solution.

At last, the pheromone trail variables will be updated using the following:

$$\tau_{ij}^{new} = \max\left(\tau_{min}, \min\left(\tau_{max}, \rho\tau_{ij}^{old} + \delta * \tau_{max} * x_{ij}'\right)\right), \forall i \in I, j \in J$$

IV. CONCLUSION AND FUTURE APPLICATIONS

This paper studied the VS framework for solving the load balance problem in a structured P2P system. The first main contribution is an effective and efficient DSLS algorithm, which leverages work in GAP. The second contribution is an in-depth analysis of the effect of capacity and workload heterogeneity on algorithm performance in both static and dynamic environments and the qualitative relationship between static and dynamic environments. We plan to investigate the following important issues in the future: We intend to explore other cost-reducing neighborhoods to further improve the DSLS algorithm. As the variance of a VS workload has a significant impact on the success ratio performance, we plan to investigate VS merging and splitting strategies to enhance the performance of the algorithms. We also plan to investigate distributed protocols to implement the proposed node registration policies.

V. REFERENCES

[1] C. Chyouhwa, T. Kun-Cheng, "The Server Reassignment Problem for Load Balancing In Structured P2P Systems, "IEEE Trans. Parallel and Distributed Systems, vol. 19, no. 2, Feb. 2008.

[2] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications," Proc. ACM SIGCOMM '01, pp. 149-160, 2001.

[3] F. Dabek, M. Kaashoek, D. Karger, D. Morris, and I. Stoica, "Wide- Area Cooperative Storage with CFS, " Proc.18th ACM Symp. Operating Systems Principles (SOSP '01), pp. 202-215, Oct. 2001.

[4] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems, " Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03), Feb. 2003.

[5] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica, "Load Balancing in Structured P2P Systems, " Proc. Second Int'l Workshop Peer-to-Peer Systems (IPTPS '03), Feb. 2003.

[6] M.L. Fisher, R. Jaikumar, and L.N. Van Wassenhove, "A Multiplier Adjustment Method for the Generalized Assignment Problem," Management Science, vol. 32, no. 9.

[7] H.R. Lourenco and D. Serra, "Adaptive Search Heuristics for the Generalized Assignment Problem," Mathware and Soft Computing, vol. 9, pp. 209-234, 2002.