

Exploiting Correlation for Query Resolution in Multi-attribute Structured Peer-to-Peer Networks

Shibayan Chatterjee

Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado, 80521
email: shibayanms@gmail.com

Anura P. Jayasumana

Electrical and Computer Engineering
Colorado State University
Fort Collins, Colorado, 80521
email: Anura.Jayasumana@colostate.edu

Abstract—A novel structured P2P architecture is presented for multi-attribute query search. The architecture uses a Ring-of-Rings topology, combined with a Bloom-filter based resolution scheme and a caching strategy to efficiently resolve queries for correlated attributes. The overlapped ring mechanism allows the system to adapt to the correlations among contents. Results indicate that using caching scheme and overlapped ring architecture produces nearly 63% reduction in latency in response, 67% reduction in hop counts and 16.4% reduction in cost of communication compared to the conventional methodologies. The results obtained in our proposed methodology has logarithmic time complexity and constant space complexity.

Index Terms—P2P Networks, Correlation, p.c.f, Overlapped Ring, Localized caching, Bloom-filters.

I. INTRODUCTION

A large number of Peer-to-Peer (P2P) services have emerged in recent years covering a wide array of real-time and non-real-time applications. Video on demand, file sharing and collaborative computing [1] are among such large scale distributed applications that have resulted in a considerable increase in P2P traffic. Systems hosting such applications requires multi-attribute resources, and include formation of on-demand distributed topologies comprising of such resources of different types and capabilities [2]. Hence, look-up for these resources requires queries to get modified from single attribute to multiple attributes.

Catering to requests for resources characterized by multiple attributes requires P2P architectures to move from a single dimensional search to multi-dimensional search for resources. Also architectures that lookup multiple-attributes resources should resolve range queries in parallel, and be scalable. For example, a query to look-up hosts for web services should have 50-60 cores and T_x between 10-20Gbps. The brute force resolution technique will search for each attribute separately and requires high overhead. Although it is well known that different subsets of attributes in multi-attribute systems are often highly correlated [11], existing work treat them as independent of each other, and thus organize and search for them independently assuming orthogonal dimensions. Here by correlated, we mean that for a specific value or a small range of values in one attribute, there exists likely ranges of values for other (correlated) attributes. The correlation may be positive, negative or even non-linear. Not only are the attributes

of resources correlated, but so are the attributes specified in queries, e.g. if a query is looking for some specific value of an attribute, then there is a probable range of values for some other attribute or a set of attributes as well [11]. Moreover, with scalable systems, the applications may often need duplicate or triplicate resources to provide fault tolerance, high availability, QoS and even enhanced performance. It therefore becomes necessary to look up multiple resources which satisfy the specified ranges for different attributes. Such a lookup can be facilitated efficiently if a value or a range of values of one attribute always point to the most likely ranges of values of other correlated attributes. If the architecture supports such an arrangement, then the lookup process doesn't need to traverse the entire range of the other attributes to resolve the query. The proposed architecture achieves this.

This paper presents and evaluates an architecture to enhance multi-attribute query resolution by exploiting correlations among attributes. The goal is to reduce the latency and hop count for lookup, with less overhead and reduced communication cost. The proposed hybrid structured P2P system provides logarithmic time complexity for look up of any number of attributes and range for each attribute. A caching mechanism coupled with an overlapped ring is integrated to exploit correlations among attributes. A Bloom filter based data structure is used to minimize the communication costs. Section II discusses related work. Section III illustrates the correlation among attributes, and Section IV presents the proposed system architecture. Section V presents the performance results followed by conclusion in Section VI.

II. RELATED WORK

P2P systems uses Distributed Hash Table (DHT), for efficient data modification and lookup protocols like Chord [3] in multi-attribute query resolution[4-6]. MAAN [4], SkipMard [5], and Hyper-cube based logical graphs [6], uses DHT and structured P2P topology, to resolve multi-attribute queries. Usage of Bloom-filters for data management of multi-dimensional range queries is addressed in using methodologies like schematic indexing in BSI [7], *caching* search results of past conjunctive queries [8], and multi-keyword searching [9].

III. ATTRIBUTE CHARACTERISTICS

A multi-attribute query Q comprising of multiple attributes and their range values resembles the expression:

$$Q \equiv (a_{1q} = [l_{1q}, u_{1q}]; a_{2q} = [l_{2q}, u_{2q}]; \dots; a_{rq} = [l_{rq}, u_{rq}])$$

where, $a_{1q}, a_{2q}, \dots, a_{rq}$ are the different attributes queried for, with $(a_{1q}, a_{2q}, \dots, a_{rq}) \in A_M$. Here, $A_M \in \mathbb{Z}^+$ represents the set of attributes. The lower limits are $[l_{1q}, l_{2q}, \dots, l_{rq}]$ and the upper limits are $[u_{1q}, u_{2q}, \dots, u_{rq}]$ for different respective attributes. A set of resources (S_{res}) having resource count (R), where $R \in \mathbb{Z}^+$ is represented by the expression:

$$S_{res} \equiv (a_1 = [l_1, u_1]; \dots; a_n = [l_n, u_n]; ip = [ip_1, \dots, ip_n])$$

Here $[a_1, a_2, \dots, a_n] \in A_M$, with lower limits $[l_1, l_2, \dots, l_n] \in \mathbb{Z}^+$ and upper limits $[u_1, u_2, \dots, u_n] \in \mathbb{Z}^+$. To relate the proposed architectural features to correlations and characteristics of attributes, as well as for performance evaluation later, we use an attribute set generated by ‘ResQue’ [11], a P2P resource generation tool. Resource attribute parameters of ‘PlanetLab’ nodes, with their counts and range, mean and standard deviation of each attribute along with their correlated attributes with Pearson correlation function (p.c.f.) is summarized in are summarized in [12], Tables-4.[1-2]. To illustrate the correlation among different attributes and to find correlation values that will be used by the architecture, scatter plots of attributes are used. The range values of these attributes has a high degree of positive correlation, p.c.f greater than 0.85. Characteristic equation defining correlation of attributes namely [DSize, DFree], $[T_X, R_X]$, [1mLD, 5mLD] and [1mLD, 15mLD] w.r.t each other is obtained by linear curve-fitting (illustrated in [12], Figures-4.[3-6]) and logarithmic (illustrated in [12], Figures-5.[1-4]) on the scatter plots of the attributes. The characteristic equations of above attribute pairs are mentioned in [12], Equations-5.[1-4] respectively.

IV. SYSTEM ARCHITECTURE

To exploit correlations among the different attributes, we propose a ring-of-rings (ROR) architecture comprising of a main ring (MR), and several sub-rings (SR) connected to MRs with common nodes contributing to both MR and SR. Chord [3] protocol is followed in both MR and SR for scalability, and to resolve queries in logarithmic time. MR nodes are responsible for both individual attribute, as well as range values of each attribute, whereas SR nodes are responsible for range values of each attribute. A ROR architecture responsible for attributes, Attrb-[a,b,c,d,e] along with their range values $[0; \dots; (2^k - 1)]$ where $k \in \mathbb{Z}^+$ is illustrated in Figure-1. Each MR node, maintains a hash-table (HT_{MR}), and is aware of all other MR nodes. External queries for any number of attributes and their range values may arrive randomly to any MR node. The MR node looks at the query, parses out its queried attribute (if any) along with its range and propagates it to the SR for resolution. The remaining part of the query is propagated to other nodes in MR for other queried attributes to get resolved. The SR nodes maintains a finger table (FT) for addressing range values of the attribute, that the specific SR is responsible for. To benefit from correlation among attributes, we propose

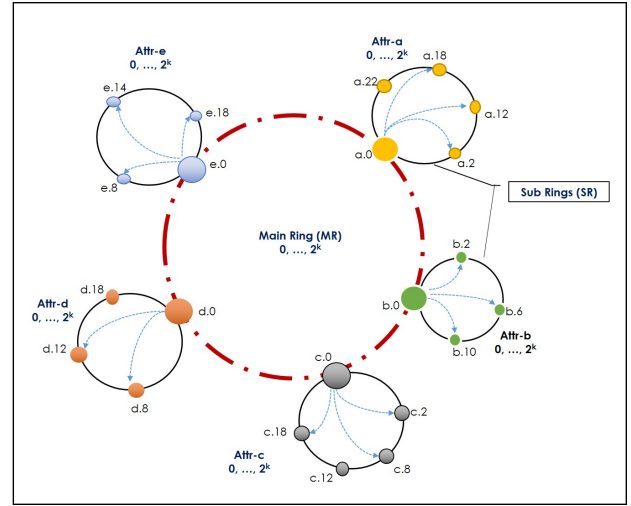


Fig. 1: ROR - System Architecture

caching scheme and overlapped ring are obtained as a result of modifying ROR architecture and integrating the system as a whole.

A. Caching Scheme

As discussed earlier, most resources in multi-attribute systems have correlated attributes. Queries for these resources can be resolved quickly by caching the range values of correlated attributes w.r.t each other within a single SR. Each SR, has a bitspace allocated $[0; \dots; (2^k - 1)]$ where $k \in \mathbb{Z}^+$. A node in SR calculate a multiplying factor (MF) as mentioned in [12], Equation-3.1, to map out the range of an attribute (say Attrb-a), the bitspace is responsible for. Range values of other attributes alongwith their resource count is summarized in [12], Table-4.2. So any node having id k_2 in SR, responsible for bitspace $[k_1; k_2]$ will also be responsible for correlated attribute having range of values $[k_1 * MF; k_2 * MF]$. It has been observed that for a range value of a single attribute there are ranges of values for other attributes with which it is correlated. An example correlated attribute pair Attrb-[a,b], has resource counts as mentioned below:

Attrb-a: a_1 ; Attrb-b: $[b_{11}, b_{12}, \dots, b_{1n}]$; Resource count: x_1 where $a_1, [b_{11}, b_{12}, \dots, b_{1n}], x_1 \in \mathbb{Z}^+$. In the above example, for a particular value of Attrb-a, say a_1 there is a range of values for Attrb-b $[b_{11}, b_{12}, \dots, b_{1n}]$. The resource count for this above mentioned correlated value pairs is x_1 . Each of these range values of Attrb-b for each individual value of Attrb-a are cached in individual nodes of SR. As a result when queries are generated for each of these attributes with range values of each attribute falling within the ranges of Attrb-a and Attrb-b it gets resolved within the same SR. This optimization saves considerable amount of overall computational time, memory overhead and it also reduces response latency. The caching architecture is illustrated in Figure-2. In Figure-2, if attributes are correlated as Attrb-(a,b), Attrb-(c,d,e), Attrb-(f,g), each of the attribute tuples is resolved within the same SR. The user generated query Q , doesn't have to propagate to a separate

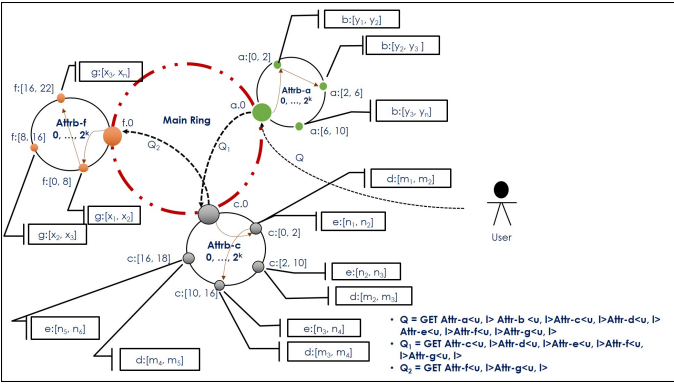


Fig. 2: Caching Architecture

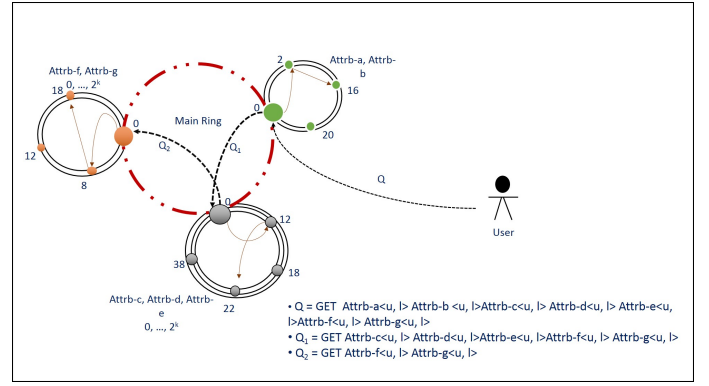


Fig. 3: Overlapped Ring Architecture

SR to get the remaining attributes resolved. The algorithm for resource selection using this methodology is presented in [12], Algorithms-5, and notations summarized in [12], Table-3.1.

B. Overlapped Ring

In overlapped ring architecture, SRs of correlated attributes are made to overlap on each other. The concept is the following. For correlated attributes it is possible to predict the range of values of one attribute from the range of values of the other attribute with which it is correlated. To achieve this use the characteristic equations mentioned earlier in Section-III. As a result, queries for two different correlated attributes with range values get resolved within the same SR. This reduces overall computational time, communication overhead and latency in response. For example, if Attrb-a , are correlated attributes, SR for both are made to overlap on each other as illustrated in [12], Figure-5.5. Similar to caching scheme, here also each SR node calculates range value of each attribute using MF from [12], Equation-3.1. The nodes contributing in SR are aware of this range values w.r.t their SR id. For example, if Attrb-a , is correlated with Attrb-b , for range values $[k_1 * MF; k_2 * MF]$ of Attrb-a , the corresponding range values in Attrb-b is found out, say $[k'_1 * MF; k'_2 * MF]$ and it's also resolved by node id k_2 . This corresponding correlated range values are obtained using the equation characterizing range values of Attrb-a , Attrb-b . Each node in SR, maintains a hash-table as mentioned below, which has range values of both attributes along with their resource IP Address.

Node ID: k_1 ; Attrb-a : $[a_{11}, \dots, a_{1m}]$; Attrb-b : $[b_{11}, \dots, b_{1m}]$; Resc. IP: $[ip_{11}, \dots, ip_{1m}]$

Node ID $k_1 \in \mathbb{Z}^+$ in SR responsible for Attrb-a , Attrb-b will have values $[a_{11}, \dots, a_{1m}] \in \mathbb{Z}^+$ for Attrb-a , values $[b_{11}, \dots, b_{1m}] \in \mathbb{Z}^+$ for Attrb-b and resource IP addresses $[ip_{11}, \dots, ip_{1m}]$ which has values in the said ranges. If a query is generated for both of these attributes, falling within the above said ranges, it gets resolved within the same node itself. The query doesn't need to propagate to a separate SR to get resolved individually. The overlapped SR architecture for correlated attributes Attrb-a , Attrb-c , Attrb-d , Attrb-e and Attrb-f is illustrated in Figure-3. The algorithm for resource selection based on correlation is presented in [12], Algorithm-9.

With the above designs integrated with ROR, random queries Q , are generated to random MR nodes (say Q_{node}). Q_{node} after receiving Q parses Q , retrieves the queried attribute range (naming it as sub-query) and sends out the remaining query of Q to the next responsible MR node. The SR nodes retrieves the list of resources corresponding to the sub-query and creates a normal Bloom-filter (nBF) [10], representing the resources. The process of Bloom-filter formation in SR is illustrated in [12], Figure-3.3. These Bloom-filters has significantly less sizes, incurs minimum computational space, and introduces less computational overhead than the long list of resources. The parameters necessary for creation of nBF are: size (m), number of resources to be represented (N) and number of Hash-Functions (k). Here, m is kept constant to perform bitwise AND operation, which counterparts the intersection operation to generate the final list of resources against Q . The Bloom-filter indices, retrieved by hashing resource IP Addresses, are set to 1 else left unchanged at 0. Each SR after creating nBF sends them to Q_{node} , which performs bitwise AND operation on all nBFs amongst each other and derives the resultant Bloom-filter (rBF). Q_{node} sends the rBF back to the respective SR nodes, as illustrated in [12], Figure-3.5. The SR nodes after receiving the rBF check for resources which pass through the rBF. After generating these list of resources within each SR, the SR nodes send it to the Q_{node} . Q_{node} on receiving all the resource lists, undergoes intersection on them and generates a final list of resources, which is sent to the user as resolution of query Q , as illustrated in [12], Figure-3.5.

V. PERFORMANCE EVALUATION

We evaluate the performance of Caching scheme and Overlapped ring based ROR architectures, and then compare these proposed architectures with a conventional ROR architectures, all having same nBF parameters. For simulating performances, resource count=1000, #. of attributes=7, #. of SR nodes=[6, 9, 12, 15, 18, 21], and queries generated=200. The variation of f_p , with 'k' for different $\frac{m}{N}$ ratio for random queries is illustrated in [12], Figures-6.[13-14] respectively. Next, random queries are generated to simulation environment, which the system is capable of resolving. The nBF size 'm'

TABLE I: Performance metrics, caching and overlapped

Metrics	μ_{cache}	σ_{cache}	$\mu_{overlap}$	$\sigma_{overlap}$
Hop-counts	36.5	4.78	37.5	4.27
Latency (msec)	235.41	27.04	248.52	20.50
Msg. size (Kbytes)	100.98	1.31	101.08	1.33
Load-per-node	15.58	4.6	17.40	5.7

used, is calculated by iterating ‘k’ with $\frac{m}{N}$ (known ‘N’), where $f_p=0$, using Equation-4.2 in [12]. The performance of caching scheme and overlapped ring architectures in terms of: a) latency in response, b) hop counts, c) message sizes exchanged and d) load per node. The variation in average hop-counts and latency in response for caching scheme and overlapped ring architectures are illustrated in Figures-4.[a-b]. With increase in number of nodes in SR, the query needs to

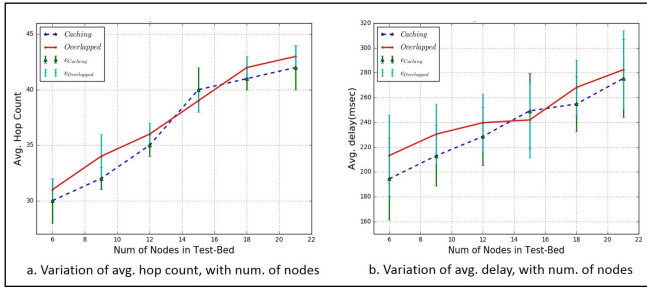


Fig. 4: Variation of avg. hop-counts, latency in response vs Num. of nodes

traverse more in each SR, causing avg. number of hops, and latency in response to increase. Queries in overlapped ring, traverses more than in caching scheme to resolve boundary values of attributes, (if any) as illustrated in Figure-4.[a]. The latency in response, in case of overlapped ring, is greater than that of caching scheme as illustrated in Figure-4.[b], since each SR node needs to compute equivalent range values for each generated query, of one attribute w.r.t another. Variation

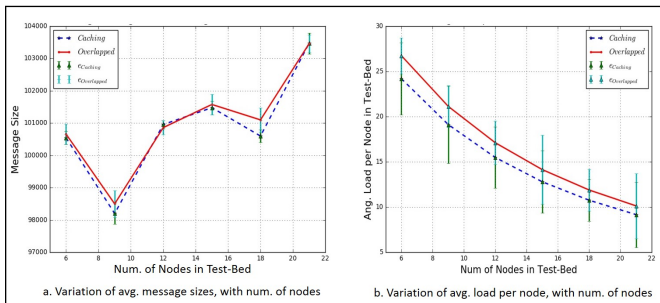


Fig. 5: Variation of message sizes exchanged, and load per node vs Num. of nodes

in average message sizes exchanged for both architectures is illustrated in Figure-5.[a]. Message sizes in case of caching scheme is less, since queries traverse less in caching scheme as compared to overlapped ring. The load per node, illustrated in Figure-5.b, decreases with increase in number of nodes

as the cumulative load per node for resolving a query gets reduced. The load per node for overlapped ring is more than caching architecture, since computation of range values in case of former is more frequent. Mean (μ) and std. dev. (σ) of each test parameter is summarized in Table-I. The results obtained from caching and overlapped ring ROR are compared with that of conventional ROR on the basis of metrics mentioned earlier. The variation of each parameter are illustrated in [12], Figures-6.[28-31]. Our findings indicate there is 67% reduction in hop-counts, 63% reduction in latency in response, and 16.4% reduction in message sizes, and 38% increase in load per node as compared to conventional ROR. The algorithmic time complexity for attribute selection in MR and SR is $\mathcal{O}(m \log_2 n)$, space complexity of messages $\mathcal{O}(m * n)$ and time-complexity of $\mathcal{O}(n)$ for Bloom-filter generation.

VI. CONCLUSION AND FUTURE WORK

A multi-attribute query resolution methodology that exploits correlations among attributes was proposed and evaluated. The architecture uses structured P2P based ROR, and Bloom-filter. We demonstrated that the proposed architecture offers scalability, resolves query for any number of attributes and any range values within logarithmic time bound and constant space bound. Simulation results show that our proposed design outperforms conventional architectures and is the most promising approach to resolve multi-attribute queries.

REFERENCES

- [1] H. M. N.D. Bandara and A.P. Jayasumana, "Collaborative applications over peer-to-peer systems—challenges and solutions," In Peer-to-Peer Networking and Applications, 6(3), 2013, 257-276.
- [2] H. M. N. D. Bandara, A. P. Jayasumana, and M. Zink, *Radar Networking in Collaborative Adaptive Sensing of Atmosphere: State of the Art and Research Challenges* Proc. IEEE Globecom Workshop on Radar and Sonar Networks, Dec. 2012.
- [3] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, *Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications* SIGCOMM01, August 27-31, 2001.
- [4] M. Cai, M. Frank, J. Chen, and P. Szekely, *MAAN: A Multi-Attribute Addressable Network for Grid Information Services* Proc. Fourth International Workshop on Grid Computing, 2003.
- [5] T. He, J. Ni, A. M. Segre, S. Wang, and B. M. Knosp, *SkipMard: A Multi-attribute Peer-to-Peer Resource Discovery Approach* Second International Multisymposium on Computer and Computational Sciences, 2007.
- [6] Y. Zhao, F.Cai, J. Xie, L. Luo, X.Teng, H. Chen, and W. Kong, *A New DHT Supporting Multi-Attribute Queries for Grid Information Services*, Proc. IEEE Int. Conf. on High Performance Computing and Communications, IEEE International Conference on Embedded and Ubiquitous Computing, 2013.
- [7] R. Dissanayaka, S. K Prasad, S. B. Navathe, and V. Goyal, *BSI: Bloom filter-based semantic indexing for unstructured P2P networks* International Journal of Peer to Peer Networks (IJP2P), Feb. 2015.
- [8] T. Ariyoshi, and S. Fujita, *Efficient Processing of Conjunctive Queries in P2P DHTs using Bloom Filter* International Symposium on Parallel and Distributed Processing with Applications, 2010.
- [9] H. Chen, H. Jin, L. Chen, Y. Liu, and L. M. Ni, *Optimizing Bloom Filter Settings in Peer-to-Peer Multikeyword Searching*, IEEE Trans. on knowledge and data engineering, pp 692-705, April 2012.
- [10] B. H. Bloom *Space/time trade-offs in hash coding with allowable errors* Commun. ACM, 13(7):422-426, 1970.
- [11] H. M. N. D. Bandara and A. P. Jayasumana, "On Characteristics and Modeling of P2P Resources with Correlated Static and Dynamic Attributes," In Proc. IEEE GLOBECOM '11, Dec. 2011.
- [12] S. Chatterjee, "Multi-Attribute Query Resolution in Structured Peer-to-Peer Networks," MS Thesis, Dec. 2016.