OID: Optimized Information Discovery using Space Filling Curves in P2P Overlay Networks

Faraz Memon*, Daniel Tiebler*, Frank Dürr*, Kurt Rothermel*, Marco Tomsu† and Peter Domschitz†
*IPVS – Distributed Systems, Universität Stuttgart †Bell Laboratories Alcatel–Lucent Stuttgart
{firstname.lastname}@ipvs.uni-stuttgart.de {firstname.lastname}@alcatel-lucent.de

Abstract

In this paper, we present the system design and evaluation of a Space Filling Curve (SFC)-based P2P information discovery system OID. The OID system uses multiple SFCs to significantly optimize the performance of multi-attribute range queries, particularly for applications with a large number of data attributes where a single big SFC-based index is inefficient. The basic idea is to have multiple SFC-based indices and select the best one to perform a query. We also introduce two tree-based query optimizations that increase the scalability of the system.

1 Introduction

Information discovery in massively distributed P2P networks has been a widely studied topic in the last decade. The research in this area has paved the way for scalable, efficient, and fault-tolerant P2P overlay networks. One such class of scalable P2P overlay networks is the Distributed Hash Tables or DHTs. Conventional DHT-based overlay networks used to support data lookup using exact match queries only. However, sophisticated P2P applications require data lookup that supports more than just exact match queries, e.g. resource discovery in grid computing relies on multi-attribute range queries. Therefore, recently DHT-based systems have been extended to support a wider range of complex queries.

A fundamental class of complex queries is multi-attribute range queries. Information discovery systems such as [2, 8, 17, 18] make use of Space Filling Curves (SFC) [3] to support multi-attribute and range queries over DHT-based P2P overlay networks. These systems have been shown to perform well for applications that have data objects with a small number of attributes. However, the performance of such systems declines significantly with increasing number of data attributes [8, 17]. In this paper, we present a SFC-based P2P information discovery system that does not suffer from performance deterioration with applications that have a large number of data attributes. Our system supports multi-attribute range queries by extensively optimizing the use of SFCs over DHT-based P2P networks.

If each attribute in a data object is considered as a dimension and the combination of attribute values is considered as coordinates, then each data element can be viewed as a point in a multi-dimensional space. A SFC is a line that passes through this space linearizing each point in it, while preserving data locality, i.e. points close to each other in the space are mapped close on the SFC line with a high probability. The SFC line can then be mapped onto a 1-dimensional P2P overlay network such as Chord [20]. Due to locality preservation, data objects that are close in the multi-dimensional space are usually mapped to sets of neighboring peers in the P2P overlay network which allows for efficient query processing.

Currently, information discovery systems use a single SFC over all the attributes in data objects. However, in typical applications, not all attributes are used at the same time to perform a query. The attributes that are not present in the query are assumed to be wildcards. A large number of wildcards in a query results in a large number of segments on the SFC line. Hence, a large number of peers have to be evaluated for the query resolution. Therefore, it becomes prohibitively expensive to use SFCs in such manner for applications that a have large number of data attributes. Another concern with SFCs is that the locality preservation starts to deteriorate as the number of dimensions increases. This is due to the fact that the distance between two points in a Euclidean space usually increases with increasing number of dimensions. This property also renders SFCs unusable for applications with a large number of data attributes.

Our system design enables the efficient use of SFCs for a large number of data attributes to perform information discovery in DHT-based P2P overlay networks. The basic idea is to use multiple SFC-based mappings, with each mapping over a small number of attributes. The best SFC-based mapping is then selected for query processing. We also introduce two query optimizations that effectively re-

Published in proceedings of 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08), pages 311-319, Melbourne, Australia, December 2008.

http://dx.doi.org/10.1109/ICPADS.2008.57

duce the total number of parallel messages in the network, and distribute the computation load of query resolution over the network. Our simulation results show that our query optimization strategies are more scalable than the previously suggested ones.

The rest of the paper is organized as follows: In Section 2 we present a review of the related work in the area. In Section 3 we discuss the architecture of the OID system in detail. Results from system evaluations are presented in Section 4. Finally, we wind up the paper with a conclusion and notion of the future work in Section 5.

2 Related Work

P2P information discovery systems supporting multi-attribute range queries can be divided into two major categories. The first category includes systems that enable such queries using specialized overlay structures. Systems that stack layers on top of a DHT-based P2P overlay network fall into the second category.

The first category of P2P information discovery systems such as DPM [1], Mercury [5] and P-Grid [7] enable multi-attribute and range queries with routing guarantees by organizing peers in a specific overlay structure. However, due to complex structural requirements, these systems incur high maintenance cost. Further, DHT-based P2P overlay networks have evolved over a period of time. As a result, scalable implementations of DHTs (e.g. OpenHash [11]) are widely available which makes it easier to extend them for information discovery.

Systems that enable multi-attribute range queries by stacking layers on top of a DHT-based P2P overlay network can be further divided into two classes. The first class includes systems that maintain a separate index on each data attribute. PHT [15] uses a trie structure to enable range queries over a single attribute on top of a DHT. MAAN [6] and Triantafillou et al. [21] use locality-preserving hashing to map the value range of an attribute on continuous nodes over the Chord ring. Andrzejak et al. [2] use Hilbert SFC [10] to map the value range of an attribute on neighboring nodes in a CAN [16] network. Similarly, Shu et al. [18] use the z-curve (aka Morton-order or z-order) [13] as a locality preserving mapping over Skip Graphs [4] to support range queries over an attribute. All these systems employ a straight-forward approach to resolve multi-attribute queries; they perform multiple queries over individual attributes and then join the results at the query originator. Such query resolution incurs high latency and network load.

The second class of layered DHT-based P2P systems maintain a single index over all data attributes. SCRAP [8] uses the z-curve over Skip Graphs to perform multi-attribute range queries. The authors of SCRAP explicitly mention the problem of declining performance of SFCs with high dimensional data. The Squid system [17] uses Hilbert SFC over a Chord ring to perform multi-attribute range queries. Their simulations also display a similar pattern of declining performance with increasing data dimensions.

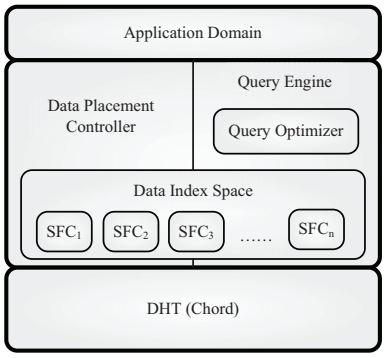

Figure 1. System Architecture

With respect to the categorization discussed above, our system is a hybrid approach to the layered DHT-based P2P systems. Similar to the first class, our system maintains multiple indices for query resolution. However, instead of having only single attribute indices, our system retains multi-attribute indices using SFCs. Each of the indices in our system is limited in dimensionality to avoid the problems discussed in Sec. 1 with large dimensional SFCs. Similar to the second class of layered DHT-based system, our system uses a single index to resolve a multi-attribute range query. The resulting main challenge is to select the appropriate index from several multi-dimensional indices to perform the query resolution.

3 System Architecture

The architecture of the OID system is a layered architecture. Fig. 1 shows the three layers of the architecture as boxes with thick borders. The top layer is the application domain layer, consisting of Internet-scale distributed applications such as resource discovery in grid computing. The middle layer is the OID framework layer that can be further classified into three components: data index space, data placement controller and the query engine. The bottom layer is the DHT layer that provides the functionality for efficient lookup of data objects.

A data object in our system is represented by a list of attribute-value pairs. For example, a data object can be defined as ($\texttt{CPU Speed} = 1.2$ GHz, $\texttt{Mem Size} = 1024$ MB, $\texttt{HDD Size} = 50$ GB, $\ldots$). Queries are defined as conjunction of tuples of the form (attribute *operator* value), where supported operators are $=, \neq, <, >, \leq$ and $\geq$, e.g. $(\texttt{CPU Speed} > 1.0) \wedge (\texttt{Mem Size} <= 512)$.

The data index space in the OID framework layer con-

sists of multiple SFC-based mappings. In order to establish the data index space, a distributed application designer identifies necessary attribute combinations (cf. Sec. 3.1). These attribute combinations are then used to define multiple SFC-based mappings (cf. Sec. 3.2). The data placement controller assigns the data objects to nodes participating in a DHT-based P2P overlay network (cf. Sec. 3.3).

A multi-attribute range query is submitted by an application to the query engine of the OID framework. The query engine then uses one of the SFC-based mappings in the data index space and the DHT protocol to route the query to the peers responsible for the query resolution (cf. Sec. 3.4). The query optimizer, which is a part of the query engine achieves efficient and scalable query performance by executing two different optimization techniques (cf. Sec. 3.5).

3.1 Attribute Domain Sub-setting

P2P information discovery systems such as [2, 8, 17, 18], use a single SFC-based mapping over all data attributes. The performance of such a mapping declines drastically when the number of data attributes is large. Therefore, we suggest that subsets of attributes are defined such that multiple SFC-based mappings could be utilized for efficient lookup.

Definition 1 *Let an attribute domain* $A = \{a_1, a_2, a_3, \ldots, a_n\}$ *be the set of attributes used to define data objects. Attribute domain sub-setting is defined as the creation of* p *sub-domains* $A_1, A_2, \ldots, A_p$ *such that* $\forall\ i, j\ \in\ [1, p]$*:* $A_i \subseteq A$*,* $A_i \neq A_j$ *for* $i \neq j$*,* $|A_i| > 1$ *and* $\bigcup_{i=1}^{p} A_i = A$*.*

Instead of defining a single SFC-based mapping for the complete attribute domain A, we define a SFC-based mapping for each $A_i \subseteq A$. Since each SFC-based mapping indexes each data object, defining a SFC-based mapping for each possible attribute sub-domain would lead to a large space requirement. Therefore, the number of attribute sub-domains is limited by the value of a parameter $p < 2^n$.

The challenge is to determine a criterion for attribute domain sub-setting such that efficient query processing is possible. We assume a P2P information discovery system in which certain queries are much more popular than the others. This assumption is realistic, since typical P2P systems have shown to exhibit skewed query popularity distribution [9, 12, 19]. Based on this assumption, we present the following heuristic-based solution.

Let $Q = \{q_1, q_2, \ldots, q_{p-1}, q_p, \ldots, q_n\}$ be the set of queries in the system with unique attribute combinations, such that $P(q_1) \geq P(q_p) \geq P(q_n)$ for $1 \leq p \leq n$. $P(q_i)$ denotes the probability of occurrence of the query q_i. Let A_{qi} be the set of attributes used in query q_i. We define p sub-domains from domain A such that $A_i = A_{qi}\ \forall i \in [1, p-1]$ and $A_p = \bigcup_{j=p}^{n} A_{qj}$. This heuristic creates $p-1$ attribute sub-domains for the most popular queries and one attribute sub-domain for the rest of the less popular queries.

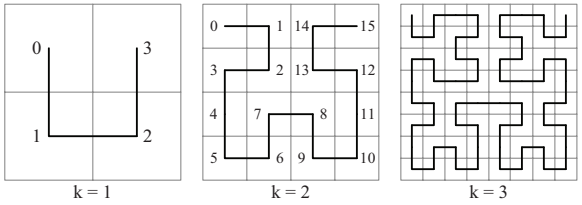

Figure 2. Hilbert SFC

Although a typical P2P application uses a significant number of attributes to define data objects, it is reasonable to expect that the number of highly popular queries would still be small. Therefore, for a typical P2P application with $10 - 15$ data attributes e.g. Edutella [14] that uses about 15 attributes to describe an e-learning resource, we expect p not to be larger than 15.

If the queries exhibit different popularities, then it is essential to define attribute sub-domains for the most popular queries because, if a popular query does not completely match an attribute sub-domain (hence, a SFC-based mapping), the accumulated overhead for such a query over a period of time, is large. Applications without the knowledge of query popularity distribution could make use of query monitoring to determine the distribution.

3.2 Defining SFC-based Mappings

The construction of a Hilbert SFC in a d-dimensional space is a two step process. The first step divides the space into smaller sub-spaces (which we call zones). The division of the space can be viewed as a recursive process where each run of the process divides the space into 2^d zones. This process continues k times resulting in $2^{k \cdot d}$ zones, where k is the approximation level. The second step involves drawing a line that passes through each of the $2^{k \cdot d}$ zones once, joining the centers of any two zones with a line segment. The center of a zone is an approximation for all the points in that zone, and the line connecting two adjacent zones imposes an order between them. The resulting SFC is a k^{th} order SFC in a d-dimensional space. Fig. 2 shows 1^{st}, 2^{nd} and 3^{rd} order Hilbert SFCs in a 2-dimensional (2D) space.

We define a SFC-based mapping for each attribute sub-domain $A_i \in A$, and each SFC-based mapping indexes each data object in the system. As an example of resource discovery in grid computing, let $A = \{$`CPU Speed`, `Mem Size`, `Busy CPU`, `Mem Used`$\}$ be the attribute domain. If two 2^{nd} order SFC-based mappings are defined for attribute sub-domains $A_1 = \{$`CPU Speed`, `Mem Size`$\}$ and $A_2 = \{$`Busy CPU`, `Mem Used`$\}$, then a data object defined as (`CPU Speed` $= 2.7\ GHz$, `Mem Size` $=$

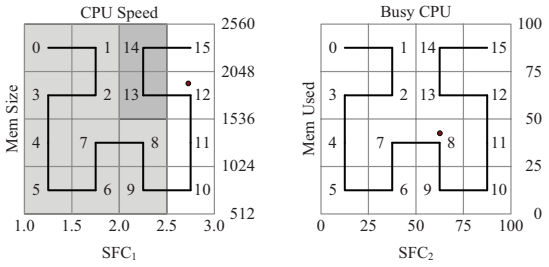

Figure 3. Data and Query Mapping on SFC

1792 MB, `Busy CPU` $= 63.3\%$, `Mem Used` $= 42\%$) would be indexed by the two SFC-based mappings as shown with a dot in Fig. 3. As a result, this data object is approximated by the 12^{th} and the 8^{th} zone on SFC_1 and SFC_2, respectively.

Queries with ranges over multiple attributes are mapped to blocks of zones, which we call clusters. A cluster is a group of continuous zones on a SFC. The number of zones that a query would be mapped to, can be easily calculated by comparing the queried range on each query axis with the corresponding axis of the SFC. For example, a multi-attribute range query defined as "(`CPU Speed` $>= 1.3$) $\wedge$ (`CPU Speed` $<= 2.3$) $\wedge$ (`Mem Size` $>= 640$) $\wedge$ (`Mem Size` $<= 2304$)", would be mapped to 12 zones on SFC_1 in Fig. 3. These zones in turn represent 2 clusters $(0-9$ and $13-14)$.

3.3 Data Placement

The SFC-based mappings effectively map the multi-dimensional data to 1-dimensional indices while preserving locality. The next step involves the use of the 1-dimensional data indices to assign the data objects to a set of peers in a P2P overlay network. We use the Chord [20] DHT to assign data objects to peers.

The Chord protocol provides the functionality for the look-up of keys in the network. Chord assigns an m bit identifier to the nodes and the keys from an identifier space in range $[0, 2^m)$. Node identifiers are generated by hashing the IP addresses of the nodes using a hash function such as SHA-1. Key identifiers are generated by hashing the keys. Chord orders the identifier space in an identifier circle modulo 2^m. A key is mapped to the node whose identifier is equal to or follows the identifier of the key in the circle [20].

Keys in our system are the data objects. Instead of using a hash function, an identifier for a data object is generated using a SFC-based mapping. The node identifiers are generated by the Chord protocol as discussed above. As an effect of the attribute domain sub-setting, the identifier space of a SFC is typically smaller than the identifier space $[0, 2^m)$ of the Chord ring. Therefore, mapping the data objects directly to the Chord ring would result in a non-uniform distribution

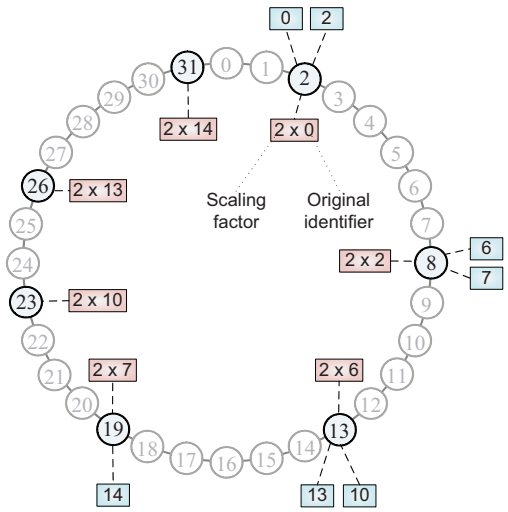

Figure 4. Chord Identifier Circle

of data objects to the Chord identifier circle. To avoid this, the identifier space of the SFC is scaled up to the identifier space of the Chord ring by a factor of $2^{m-(k \cdot d)}$.

As an example, consider a Chord ring with $m = 5$ and $N = 7$, i.e. the identifier space of the Chord ring is $[0, 32)$ with seven physical nodes as shown in Fig. 4. Physical nodes are shown as circles with bolder caption. Suppose a SFC-based mapping that assigns identifiers to the objects from an identifier space of $[0, 16)$ ($d = 2$ and $k = 2$). Without up-scaling, the data objects with identifiers $0, 2, 6, 7, 10, 13$, and 14 would be mapped to the nodes as shown with the boxes outside of the Chord ring in Fig. 4. The scaling factor is calculated as $2^{5-4} = 2$ and the same data objects with scaled up identifiers are mapped to the nodes as shown with the boxes inside of the Chord ring.

Up-scaling of the SFC identifier space to the Chord identifier space could result in a situation where some peers in the network are not responsible for any data objects. This situation occurs when the number of peers is greater than the number of identifiers in the SFC identifier space. Such a situation could be avoided by setting a relatively high approximation level for the SFC-based mappings.

3.4 Query Resolution

A query for data objects can be initiated by any peer in the network through the application layer. Once a query has been initiated, the resolution is performed in four steps. In the first step, the query initiator locally selects the best matching SFC-based mapping that has the expected lowest overhead for the query. The second step involves calculation of the zone identifiers that match the query on the selected SFC-based mapping. In the third step, the identity of the nodes that are responsible for the queried zones is deter-

mined and the query is forwarded to these nodes. Finally, the queried nodes perform a local database lookup and send the matching data objects directly to the query initiator.

When the query arrives at the OID framework layer of the query initiator, the query engine compares the attributes used in the query with the attributes used in each SFC-based mapping defined in the data index space. A query can either completely match a single SFC-based mapping or partially match more than one mappings, in terms of queried attributes. If a query completely matches a mapping, then that mapping is chosen for further processing.

For partially matching queries, the goal is to select a mapping that uses minimum number of peers to perform a lookup. To achieve this goal, global knowledge about the mapping of zones to peers would be required. However, a peer can estimate the number of peers involved in the query resolution without this global knowledge. Let N be the total number of peers in the network. Since, the SFC identifier space is uniformly distributed over these peers, the number of zones per peer for each SFC-based mapping is given by $2^{k \cdot d}/N$. Let z be the number of zones that match the queried ranges on a SFC-based mapping. Then, the number of peers involved in the query is approximated by $(z/2^{k \cdot d}) * N$. Since the total number of peers is the same for each SFC-based mapping, the partially matching mapping with the smallest value for the fraction $z/2^{k \cdot d}$ (zone fraction) is chosen for further processing. If the zone fraction is equal, one of the mappings is chosen randomly.

The next step involves calculation of zone identifiers that match the query. If the submitted query completely matches a SFC-based mapping and is a multi-attribute point query, the calculation results in a single zone identifier. Querying the peer that is responsible for this zone identifier would resolve the query. For queries involving ranges of values or wildcards, the zone identifier calculation usually results in more than one cluster with each cluster containing one or more zones.

The basic approach to resolve a query that results in more than one cluster is as follows. First, a DHT lookup is performed at the query initiator for the first zone in each cluster. The DHT lookup determines the identity of the peer responsible for this zone. The query along with the cluster is forwarded to the responsible peer. If all the zones in the cluster are resolved at this peer, then the cluster is completely resolved. Otherwise, this peer forwards the query to the peer responsible for the first zone in the remaining unresolved part of the cluster. This process continues until all the clusters are completely resolved.

3.5 Query Optimization

Typically, a multi-attribute range query results in a large number of zones (and therefore clusters), whose identifiers have to be calculated. Performing this calculation at the query initiator takes significant time. Moreover, a DHT lookup for each cluster would result in extremely high message load at the query initiator. A peer is usually responsible for a large number of query clusters. Therefore, if the basic query resolution approach is used, the same peer could receive the same query multiple times. To avoid these drawbacks, we introduce the following two query optimizations.

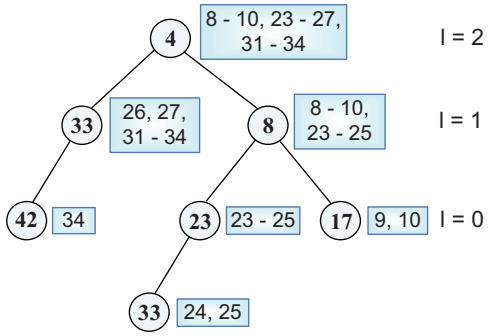

Figure 5. Routing Optimization

3.5.1 Routing Optimization

The first optimization reduces the parallel number of messages in the network and the outgoing message load at the query initiator. The basic idea is to embed a distribution tree in the network to limit the number of messages a peer has to send i.e. fan-out of a node. The total number of parallel messages in the network can be limited by limiting the depth-level of the distribution tree.

Let parameters f and l be the fan-out and the depth-level, respectively. After the zone identifiers are calculated at the query initiator, they are ordered in an ascending order and divided equally into f buckets. A DHT lookup is initiated for the first zone of each bucket. Once the peer responsible for this zone is identified, the bucket along with the query, the identity of the initiator node and the values of f and l, are forwarded to this peer.

When a peer receives a bucket from some other peer in the network, it decreases the value of l by 1 and removes the zone identifiers that it is responsible for from the received bucket. The peer then performs a local database lookup. The matching data objects are directly transferred to the query initiator. If the value of l is greater than zero, the remaining zones in the bucket are further divided into f smaller buckets and each bucket is forwarded to its responsible peer. If l is zero, the remaining bucket is not divided any further and is forwarded to the peer responsible for the first zone in the bucket. In this case, the query forwarding process continues sequentially without increasing the parallel number of messages until all buckets are empty.

Consider the Chord ring with $m = 6$ and $N = 10$. The ten physical peers have identifiers 1, 4, 8, 17, 23, 33, 42, 58, 62 and 64. Suppose that a query is initiated at the peer 4 for clusters '$8-10$', '$23-27$', and '$31-34$', with parameters $f = l = 2$. Fig. 5 shows the successive query resolution. Peer 4 divides the list of zones into two buckets. The bucket containing the zone identifiers '$8-10, 23-25$' is transmitted to peer 8, because it is responsible for the first zone (8) in the bucket. The responsible peer is determined by performing a DHT lookup which has been omitted from the figure for simplicity. Similarly, the bucket containing the identifiers '$26, 27, 31-34$' is transmitted to peer 33.

Peer 8 performs a local database lookup, removes zone 8 from the received bucket, divides the bucket further into two buckets, and transfers them to peers 17 and 23. Equal division of the bucket at peer 8 is not possible, therefore the bucket sent to peer 23 has one more item than the one sent to peer 17, as the algorithm tries to assign continuous zones to the same buckets. The algorithm continues in a similar manner at each peer until the value of parameter l becomes zero, and there are still some unresolved zones in a bucket. This situation arises at peer 23. Therefore, the bucket at peer 23 remains unsplit and is forwarded to peer 33.

Compared to the basic query resolution approach, the routing optimization increases the query resolution latency because fewer messages are processed in parallel. But due to the reduction in the network load, scalability is increased.

3.5.2 Computation Load Distribution

The second query optimization reduces the computation effort of calculating the zone identifiers at the query initiator. In the worst case, the query initiator calculates $O(2^{k \cdot d})$ zone identifiers, i.e. the number of zone identifier increases exponentially with d and k. The value of k not expected to be smaller than 8 to achieve fine grained division of the SFC. The basic idea of this optimization is to distribute the computation effort of calculating the zone identifiers over several peers in the network. Once the calculation is finished, the peers that hold the calculated zone identifiers initiate the distributed query resolution using the routing optimization.

A similar query optimization has been implemented by the Squid [17] system. Although their optimization reduces the number of peers that are involved in the query resolution, it does not scale for popular queries in the system. In the Squid system, if a query has a high queried frequency, then the peers that perform the identifier calculations would become heavily loaded with computation tasks.

In order to distribute the computation load, we assume that each peer in the network defines a parameter $k_p \geq 1$ for each SFC-based mapping. k_p denotes that the maximum number of zone identifiers a peer is willing to calculate is $2^{k_p \cdot d}$. A peer first refines a query for at least one approximation level to produce zone identifiers. The peer then checks whether performing the refinement for the next approximation level would exceed the threshold of $2^{k_p \cdot d}$. If the threshold is not exceeded the peer refines the query for the next approximation level. If the threshold would be exceeded, the peer divides the identifiers into f buckets, selects f random peers in the network, and transfers the buckets to these peers. The parameter l works in the similar manner as discussed in the previous section. The process continues until the zone identifiers for the maximum approximation level k are obtained. After that, the peers that hold the final zone identifiers initiate the lookup for data objects using the routing optimization.

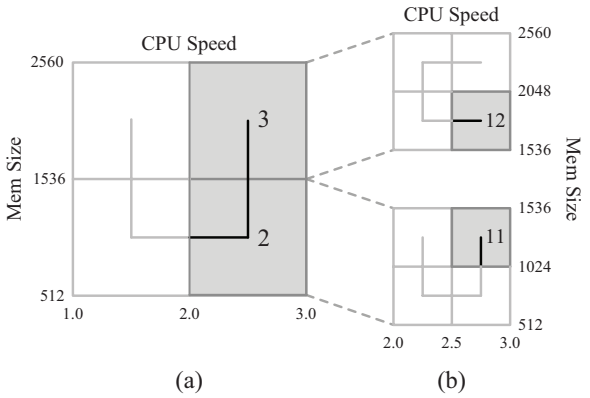

Figure 6. Successive Refinement of a Query

For a query defined as "(`CPU Speed` $>=$ 2.7) $\wedge$ (`CPU Speed` $<=$ 3.0) $\wedge$ (`Mem Size` $>=$ 1024) $\wedge$ (`Mem Size` $<=$ 1891)" and with parameters $k = 2$, $k_p = 1$ and $f = 2$, the query initiator calculates zone identifiers 2 and 3 on the 2-dimensional SFC shown in Fig. 6 (a). The identifiers for only these two zones are calculated because others do not match the queried range. The initiator then divides these identifier into two buckets and transfers them to two random peers in the network. One of the random peer then refines zone 2 to produce zone identifier 11 and the other peer refines zone 3 to produce zone identifier 12 (Fig. 6 (b)) for next approximation level.

For computation load distribution, each peer involved in calculating the zone identifiers has a load of $O(2^{k_p \cdot d})$. Computation load distribution also reduces the average size of the messages in the network, compared to the routing optimization. Since a peer computes only a limited amount of zone identifiers for a query, the number of clusters transmitted in a single message is therefore low.

4 System Evaluation

In this section, we present experimental evaluation of several aspects of our system, using simulations. We have

Param	Value(s)	Description
N	$10^2, 10^{2.5}, 10^3 \ldots 10^5$	# of peers
O	$10 * N$	# of data objects
f	10	fanout
l	$\lfloor log(N) \rfloor - 1$	depth-level
k	8	k for $SFC_{1,2,3}$
k_{p1}	8	k_p for SFC_1
k_{p2}	5	k_p for SFC_2
k_{p3}	4	k_p for SFC_3

Table 1. Evaluation Parameters

implemented Hilbert SFC-based mappings over the Chord DHT to perform these simulations.

We consider the use case of resource discovery in grid computing. An attribute domain $A = \{$`CPU Speed`, `Mem Size`, `Busy CPU`, `HDD Free`$\}$ is used to describe resources as data objects in our system. We assume that the following three attribute sub-domains have been defined by the system administrator, using the attribute domain sub-setting discussed in Sec. 3.1: $A_1 = \{$`CPU Speed`, `HDD Free`$\}$, $A_2 = \{$`CPU Speed`, `Busy CPU`, `Mem Size`$\}$ and $A_3 = \{$`CPU Speed`, `Mem Size`, `Busy CPU`, `HDD Free`$\}$. Note that the attribute sub-domain A_3 is equal to the complete attribute domain A. We define the following three SFC-based mappings: SFC_1, SFC_2, and SFC_3 corresponding to A_1, A_2, and A_3 respectively.

4.1 Evaluating SFC-based Mappings

In this section, we show that the best performance for multi-attribute range queries is achieved over corresponding completely matching SFC-based mappings. Therefore, having a single large SFC is not sufficient. We also show that our system chooses the optimal SFC for query processing in case the query partially matches more than one SFCs.

For evaluating the performance of the SFC-based mappings defined above, we perform 2D and 3D queries over them using the simulation parameters shown in Table 1. The values for f and l are chosen such that the number of parallel messages are restricted to 10% of the network size in the worst case. We increase the number of data objects with the increasing network size, just like in a typical P2P system. The following performance metrics are measured:

Total Hops – Total number of hops of all messages needed to resolve a query.

Processing Peers – Number of peers that perform a local database lookup to evaluate the query.

Data Peers – Number of peers containing the data objects that match the query (subset of processing peers).

We perform a 2D range query that has the same attributes as SFC_1 and a 3D range query that has the same attributes as SFC_2, over all three SFCs. For each network size, the query is performed 10 times starting at a random peer in the network. The metrics total hops, processing peers, and data peers are then averaged over the 10 runs. The performance of the 2D query can be seen in Fig. 7 (a)-(c). Fig. 7 (d)-(f) show the performance of the 3D query.

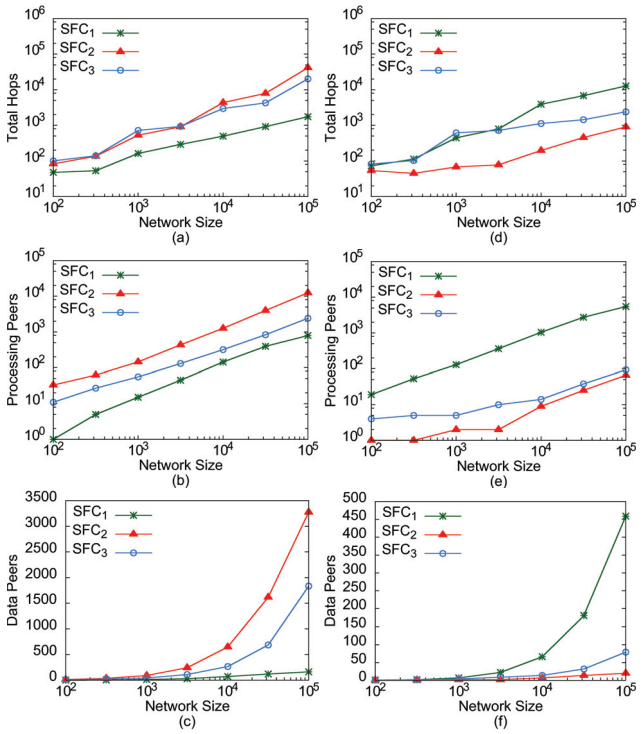

Figure 7. Performance of 2D and 3D Queries

The 2D range query performs best on SFC_1 and the 3D range query shows best performance on SFC_2 in terms of all three performance metrics. This is due to the fact that the 2D query has the same attributes as SFC_1, and the same query has to be performed with two wildcard attributes on SFC_2 and one wildcard attribute on SFC_3. Similarly, the 3D query matches SFC_2 completely and has to be performed with wildcards on the other two SFCs.

Fig. 7 (a)-(c) also show that with the large network size, the 2D query becomes prohibitively expensive on SFC_2 because of the wildcards. Similar is the case for 3D query on SFC_1 (Fig. 7 (d)-(f)). Therefore, it is important to perform a query using a SFC-based mapping that involves a minimum number of wildcards.

If SFC_1 was not defined, then the 2D query would partially match SFC_2 as well as SFC_3. The OID system would choose SFC_3 for processing the query because the zone fraction is 0.0152 for SFC_3, compared to 0.1055 for SFC_2. Similarly, if SFC_2 was not defined, SFC_3 would be chosen for processing the 3D query because the

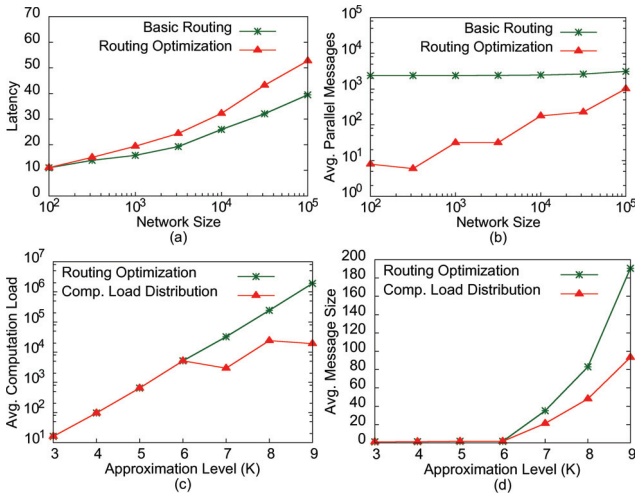

Figure 8. Query Optimizations

zone fraction is 0.00042 for SFC_3, compared to 0.1055 for SFC_1. In both cases, the best matching SFCs would be chosen for the query resolution.

4.2 Evaluating Query Optimizations

In this section, we compare the performance of the query optimization strategies discussed earlier.

4.2.1 Basic Routing vs. Routing Optimization

In this section, we show that the routing optimization algorithm reduces the total number of parallel messages in the network with a cost of little increased latency.

For comparison of the basic query routing strategy (Sec. 3.4) with the routing optimization (Sec. 3.5.1), we perform a 2D range query using attributes "`CPU Speed`" and "`Mem Size`" over SFC_2. The simulation parameters used for this experiment are the same as in Table 1 (excluding k_{pi}). Following performance metrics are measured:

Latency – Number of hops of the longest message path.

Average Number of parallel messages – Total number of forwarded messages divided by the number of levels of the search tree.

The performance of the two routing algorithms can be seen in Fig. 8 (a)-(b). The query resolution latency increases with the network size in case of both routing algorithms, because the number of data peers increases (Fig. 8 (a)). As expected, the routing optimization has a higher latency than the basic approach but the difference is not major. The number of parallel messages also increase with the network size for both approaches (Fig. 8 (b)). Even though the increase is significant for the routing optimization, the algorithm still produces fewer parallel messages per level compared to the basic approach, even with the network size of 10^5.

Param	Value(s)	Description
N	10^3	# of peers
O	$10 * N$	# of data objects
f	10	fanout
l	$\lfloor log(N) \rfloor - 1$	depth-level
k	$3, 4, 5, \ldots 9$	k for the SFC
k_p	3	k_p for the SFC

Table 2. Evaluation Parameters

4.2.2 Routing Optimization vs. Computation Load Distribution

In this section, we show that the computation load of calculating zone identifiers increases exponentially with increasing approximation level k. We also show that the computation load distribution algorithm reduces the amount of computation performed by a single peer. Due to the computation load distribution, the average message size is also reduced.

For comparison of the routing optimization (Sec. 3.5.1) with the computation load distribution (Sec. 3.5.2), we perform a 2D range query using attributes "`CPU Speed`" and "`Mem Size`" over SFC_2. The simulation parameters used for this experiment are shown in Table 2. The following performance metrics are calculated:

Average Computation Load per Peer – Total number of computed zones divided by the total number of peers performing the computation.

Average Message Size – Total number of forwarded clusters divided by the total number of forwarded messages.

The performance comparison of the routing optimization with the computation load distribution can be seen in Fig. 8 (c)-(d). Note that the y-axis of Fig. 8 (c) is a log scale.

The average computation load of peers increases with the approximation level (Fig. 8 (c)) with or without computation load distribution, because more zone identifiers have to be calculated. However, the computation load distribution strategy reduces this load significantly, in particular for larger values of k. For example, for $k = 9$, the average computation load of a peer is reduced by 99%.

In case of computation load distribution, the same number of peers are able to calculate all zone identifiers for $k = 3 - 5$. For $k = 6$, more peers are involved in calculation, therefore the average load goes down. Similar is the case for $k = 6 - 8$.

The average message size also increases with the approximation level (Fig. 8 (d)) in case of both optimizations, because more clusters are forwarded per message. But the computation load distribution achieves reduced message size for large values of k compared to the routing optimiza-

tion. For example, for $k = 9$, the average message size of a peer is reduced by 51%.

5 Conclusion and Future Work

In this paper, we presented the system design and experimental evaluation of the optimized information discovery system (OID), based on Space Filling Curves. Our system significantly improves the performance of multi-attribute range queries, particularly for applications with large number of data attributes where a single big SFC is inefficient. The optimized performance for such queries is achieved by defining multiple SFC-based mappings and later selecting the best one for query processing. OID chooses the best SFC-based mapping for query processing by estimating the number of peers the query would be evaluated at, for each SFC-based mapping. We also introduced two types of query optimizations that improve the system scalability. The routing optimization limits the message forwarding load of a peer and the number of parallel messages at a time in the system. The computation load distribution algorithm distributes the computation load for the query resolution over several peers in order to avoid bottleneck at particular peers.

For the first prototype of our system, we presented a criterion for defining multiple SFC-based mappings based on query popularity distribution where some queries are more popular than others. A more sophisticated algorithm that works for any kind of popularity distribution is a part of the future work.

Acknowledgements

Special thanks to Gerald Koch for helpful comments.

References

[1] R. Ahmed and R. Boutaba. Distributed Pattern Matching: A Key to Flexible and Efficient P2P Search. *IEEE Journal on Selected Areas in Communications*, 25, 2007.

[2] A. Andrzejak and Z. Xu. Scalable, Efficient Range Queries for Grid Information Services. In *Proc. of the 2nd Intl. Conf. on P2P Computing*. IEEE Computer Society, 2002.

[3] T. Asano, D. Ranjan, T. Roos, E. Welzl, and P. Widmayer. Space Filling Curves and Their Use in the Design of Geometric Data Structures. In *Proc. of the 2nd Latin American Symposium on Theoretical Informatics*. Springer, 1995.

[4] J. Aspnes and G. Shah. Skip Graphs. *ACM Transactions on Algorithms*, 3, 2007.

[5] A. Bharambe, M. Agrawal, and S. Seshan. Mercury: Supporting Scalable Multi-Attribute Range Queries. In *Proc. of the SIGCOMM Symposium on Communications Architectures and Protocols*. ACM, 2004.

[6] M. Cai, M. Frank, J. Chen, and P. Szekely. MAAN: A Multi-Attribute Addressable Network for Grid Information Services. In *Proc. of the 4th Intl. Workshop on Grid Computing*. IEEE Computer Society, 2003.

[7] A. Datta, M. Hauswirth, R. John, R. Schmidt, and K. Aberer. Range Queries in Trie-Structured Overlays. In *Proc. of the 5th Intl. Conf. on P2P Computing*. IEEE Computer Society, 2005.

[8] P. Ganesan, B. Yang, and H. Garcia-Molina. One Torus to Rule Them All: Multi-dimensional Queries in P2P Systems. In *Proc. of the 7th Intl. Workshop on the Web and Databases*. ACM, 2004.

[9] A. S. Gish, Y. Shavitt, and T. Tankel. Geographical Statistics and Characteristics of P2P query strings. In *Proc. of 6rd Intl. Workshop on P2P Systems*, 2007.

[10] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. In *Mathematische Annalen*, 1891.

[11] B. Karp, S. Ratnasamy, S. Rhea, and S. Shenker. Spurring Adoption of DHTs with OpenHash, a Public DHT Service. In *Proc. of the 3rd Intl. Workshop on P2P Systems*. Springer, 2004.

[12] A. Klemm, C. Lindemann, M. K. Vernon, and O. P. Waldhorst. Characterizing the Query Behavior in Peer-to-Peer File Sharing Systems. In *Proc. of the 4th SIGCOMM Conf. on Internet Measurement*. ACM, 2004.

[13] J. A. Orenstein and T. H. Merrett. A Class of Data Structures for Associative Searching. In *Proc. of the 3rd SIGACT-SIGMOD Symposium on Principles of Database Systems*. ACM, 1984.

[14] C. Qu, W. Nejdl, and H. Schinzel. Integrating Schema-specific Native XML Repositories into a RDF-based e-learning P2P Network. In *Proc. of the 2002 Intl. Conf. on Dublin Core and Metadata Applications*. Dublin Core Metadata Initiative, 2002.

[15] S. Ramabhadran, J. Hellerstein, S. Ratnasamy, and S. Shenker. Prefix Hash Tree - An Indexing Data Structure over Distributed Hash Tables. In *23rd Annual SIGACT-SIGOPS Symposium on Principles of Distributed Computing*. ACM, 2004.

[16] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-addressable Network. In *Proc. of the 2001 Conf. on Applications, Technologies, Architectures, and Protocols for Computer Communications*. ACM, 2001.

[17] C. Schmidt and M. Parashar. Flexible Information Discovery in Decentralized Distributed Systems. In *Proc. of the 12th Intl. Symposium on High Performance Distributed Computing*. IEEE Computer Society, 2003.

[18] Y. Shu, B. C. Ooi, K.-L. Tan, and A. Zhou. Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems. In *Proc. of the 5th Intl. Conf. on P2P Computing*. IEEE Computer Society, 2005.

[19] K. Sripanidkulchai. The Popularity of Gnutella Queries and its Implications on Scalability, 2001.

[20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of the 2001 SIGCOMM Conf.* ACM, 2001.

[21] P. Triantafillou and T. Pitoura. Towards a Unifying Framework for Complex Query Processing over Structured Peer-to-Peer Data Networks. In *Proc. of 6th Intl. Workshop on Databases, Information Systems and P2P Computing*. Springer, 2003.