SPACE FILLING CURVE BASED MULTIDIMENSIONAL RANGE QUERIES IN PEER TO PEER GRID ENVIRONMENT

A DISSERTATION

Submitted in partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

AMIT SHARMA

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2011

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled "**SPACE FILLING CURVE BASED MULTIDIMENSIONAL RANGE QUERIES IN PEER TO PEER GRID ENVIRONMENT**" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology** in **Computer Science and Engineering** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2010 to June 2011, under the guidance of **Dr. Padam Kumar, Professor,** Department of Electronics and Computer Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: 28-06-2011

Amit Sharma

Place: Roorkee

(AMIT SHARMA)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 28/6/11

[illegible]

Place: Roorkee

(Dr. PADAM KUMAR)

Professor

Department of Electronics and Computer Engineering

IIT Roorkee

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my heartfelt gratitude to my guide and mentor **Dr. Padam Kumar**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for his invaluable advices, guidance, and encouragement for sharing his broad knowledge. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. He has been very generous in providing the necessary resources to carry out my research. He is an inspiring teacher, a great advisor, and most importantly a nice person.

I am thankful for the useful comments and suggestions from faculty members of our institute. I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample moral supports and valuable suggestions.

On a personal note, I owe everything to the Almighty and my parents. The support which I enjoyed from my father, mother and other family members provided me the mental support I needed.

AMIT SHARMA

ABSTRACT

In Grid Computing Resource Discovery is an important activity as grid scheduler needs to find the available resources for carrying out the job. Traditionally in grid environment centralized or hierarchical resource discovery mechanism are used. But this is not scalable if number of entities participating in the grid becomes very large. In recent years peer to peer based resource discovery mechanisms for grid have been widely explored. Peer to peer resource discovery mechanism evolves from file sharing mechanism on internet. They are efficient for single dimensional exact match query. But in grid computing queries are generally multidimensional range queries.

In past few years a large number of approaches have been proposed for multidimensional range queries for peer to peer environment.

Space filling Curve (SFC) are used in database to map multidimensional data into one dimensional data.

In this dissertation work we evaluate the work of Schmidt [4]. They have presented a SFC based mapping over CHORD [1] overlay network for complex queries in peer to peer environment. They have used Hilbert SFC for their experiments.

In this dissertation work we are evaluating feasibility of this system in grid environment. We are using Hilbert, Z-order and compact Hilbert SFC. Hamilton et al. [5] proposed the compact Hilbert indices where precision of various attribute differs, so that only minimum bits are used in construction of Hilbert indexes.

In the experiments we find that Compact Hilbert index is better than Hilbert and Z-order SFC but for more than four dimensions its calculation becomes very time consuming and also using SFCs in high dimension generate a large number of clusters and locality properties may not be preserved. So we are dividing the attributes into groups and query the resources from these groups and finally take the intersection of results obtained. In the simulation we find that this gives better performance than using Single space filling curve.

Table of contents

LIST OF FIGURES

Figure No. **Page No.**

LIST OF TABLES

Table No. **Page No.**

Chapter 1

Introduction and Statement of the Problem

1.1 Introduction and motivation

Grid computing provides the basic infrastructure which is needed for the sharing of resources including desktops, computational clusters, supercomputers, storage. Some problems like high energy physics, protein folding, financial modeling, and earthquake simulation have high requirement of computational power which computational grid can offer. Grid can be of various types like Desktop Grid, Cluster Grid, Data Grid, HPC (high performance computing) grid.

Desktop grid utilizes the personal computer's processing power for the computational purpose. But in this type of grid number of entities (peers/nodes) contributed to the grid can become quite large.

An application/user submits its requirement to the grid scheduler. Grid scheduler first needs to find the resources for the task that exist in the grid.

Traditionally centralized approaches are used in the grid computing where computational entities (peers) register their resources to the centralized server and scheduler can find the resource available from the server.

But in grids many resources (like available processing power, free memory) are dynamic in nature i.e. their availability keeps on changing. They need to be updated at centralized server.

So if number of computational entities becomes high as it can become in case of desktop grids centralize or hierarchical approaches for indexing the resources may not perform well. These approaches become bottleneck because of congestion, lack of computational power at server, single point of failure. Even hierarchical approaches also have similar problems as that of centralized system and do not scale well.

So in the recent years Peer to Peer Approaches for information discovery for grid computing have been proposed. Key characteristics of these systems are decentralization, scalability, dynamism and fault-tolerance. Peer to Peer networks are very popular on internet for file sharing. Peer to Peer systems like Chord [1], CAN [2], Pastry [3] and other support only the exact match query. Chord is a peer to peer overlay network proposed by researchers at MIT. We can store key/value pairs at the peers in the network. For a given key we can query the corresponding value in the network with complexity O(log n) . It is very scalable and fault tolerant.

But in grid computing the queries are generally multidimensional range queries like user want the resource value between certain ranges.

In past few years considerable research work has been done in the field of multidimensional range queries in the P2P Environment.

Some researchers have used the spatial indexes like R-Tree, KD Tree, Range search Tree, Segment Tree in peer to peer environment and others have used the space filling curves (SFC) for mapping multidimensional data into one dimension and using the query algorithm to query the one dimensional data.

In this dissertation work we review some of the work done in this field. Then we will take the SFC based approach specifically that one described the Squid system [4].

Space filling curves find application in the database (in Oracle 1995 and Transbase 2000) for dimensionality reduction.

We are using the Space filing curves like Hilbert, Compact Hilbert and Z-order curve over the Chord peer to peer overlay network.

1.2 Statement of Problem

The aim of this dissertation work is to evaluate the Space filling curves (Hilbert, Z order and compact Hilbert indexes) in Squid system [4]. For this I have implemented some of features of Squid system like mapping multidimensional data to one dimension using space filling curves (SFC) and querying over the Chord overlay network. I have introduced following features in the Squid System.

1. Use of compact Hilbert indexes [5] which can be used when ranges of attributes used in multidimensional range queries are unequal.
2. An alternative query mechanism which is different from given in the Squid System.

3. Use of Z order indexes, Squid system uses only Hilbert indexes.
4. Comparison of Hilbert, Compact Hilbert and Z-order indexes and their performance analysis.
5. Use multiple SFC mapping over P2P overlay for querying over more than 6 dimensions.

1.3 Organization of Report

Chapter 2 gives the survey of various resource discovery mechanism proposed in the past years.

Chapter 3 gives the overview of peer to peer information discovery system. It includes the working of DHT (Distributed hash table) and working of structured overlay (CHORD).

Chapter 4 gives the introduction of space filling curve (SFC) for mapping multidimensional data into one dimension. It gives working of Hilbert order, Z order curve and introduce compact Hilbert order.

Chapter 5 gives the working of Squid System which we are using in this dissertation work.

Chapter 6 gives the implementation details

Chapter 7 gives results and analysis part

Chapter 8 concludes the dissertation work and gives suggestions of future work.

Chapter 2

Background and Literature Review

2.1 Grid resource query

In general computing resources have two types of attributed

1. Static Attributes such as types of operating system installed, network bandwidth, processor speed and storage capacity (including physical and secondary memory)
2. Dynamic attribute such as processor utilization, physical memory utilization, free secondary memory size, price and network bandwidth utilization.

In general, the queries can be abstracted as lookup for objects based on a single dimension or multiple dimensions. Since a grid resource is identified by more than one attribute, An Resource lookup query or Resource update Query is always d-dimensional. Queries can specify different kinds of constraint on the attribute values. If query specifies a fixed value for each attribute, it is referred to as a d-dimensional point query. However if the query specifies a range of values for attribute, it is referred to as a d-dimensional range query. For example in range query range values for attributes are specified. For example Architecture = 'Macintosh' and type = 'Cluster' and '1GHz' <= CPU-Speed <= '3GHz' and '512 MB' <= RAM <= '1GB'

2.2 Resource Discovery in Grid [6]

Grid resources are managed by their local resource management systems (LRMS) such as Condor, Portable Bath System (PBS), Sun Grid Engine (SGE). LRMS manage job queues and initiate and monitor their execution.

Traditionally superschedulers including Nimrod-G, Condor-G and Tycoon used centralized information services such as R GMA, Hawkeye, GMD and MDS-1 to index resource information. Under centralized organization, the superschedulers send resource queries to a centralized resource indexing service. Similarly, the resource providers update the resources status periodic intervals using resource update messages. This approach has several design issues such as

1. Highly prone to single point of failure
2. Lacks scalability
3. High network communication cost at links leading to information server. This may lead to network bottleneck and congestion.
4. The machine running the information services might lack the required computational power to serve a large number of resource queries and updates.

To overcome the above shortcomings of centralized approaches, a hierarchical organization of information services has been proposed in systems such as MDS-3 and Ganglia. MDS-3 organizes virtual organization (VO) specific information directories in a hierarchy. A VO includes a set of institutions that agree on common resource sharing policies. Every VO in grid designated a machine that hosts the information services. A similar approach has been followed in the Ganglia system, which is designed for monitoring resources status within a federated of clusters. Each cluster designates a node as a representative to the federated monitoring system. This node is responsible for reporting cluster status to the federation. However this approach also has similar problems to the centralized approach such as one point of failure and does not scale well for large number of users and providers.

Recently proposals for decentralizing a Grid Resource Information Service (GRIS) have been explored. The decentralization of GRIS can overcome the issue related to current centralized and hierarchical resource indexing. A distributed system configuration is considered as decentralized of none of the participants in the system is more important than others, in case one of participant fails, it is neither more or less harmful to system than the failure of any other participant in the system. An early proposal for decentralizing grid information services was made by Iamnitchi and Foster. They proposed that every VO maintains its information services and make it available as part of P2P based network. Application Schedulers in various VOs initiate a resource lookup query that is forwarded in the P2P network using flooding (an approach similar to one applied in the unstructured P2P network Gnutellla). But this approach has a large volume of network message generated due to flooding. To avoid this, a time to live (TTL) field is associated with every message, so the peer stops forwarding a query message once the TTL expires). To an extent this approach can limit the network message traffic, but the proposed approach cannot guarantee to find the desired resource even though it exists in the network.

Recently organizing a GRIS over structured P2P networks has been widely explored.

2.3 Survey of Multidimensional range query approaches in P2P Environment

In past years various approaches of Multidimensional range query in Peer to Peer grid environments are proposed which are described briefly as below.

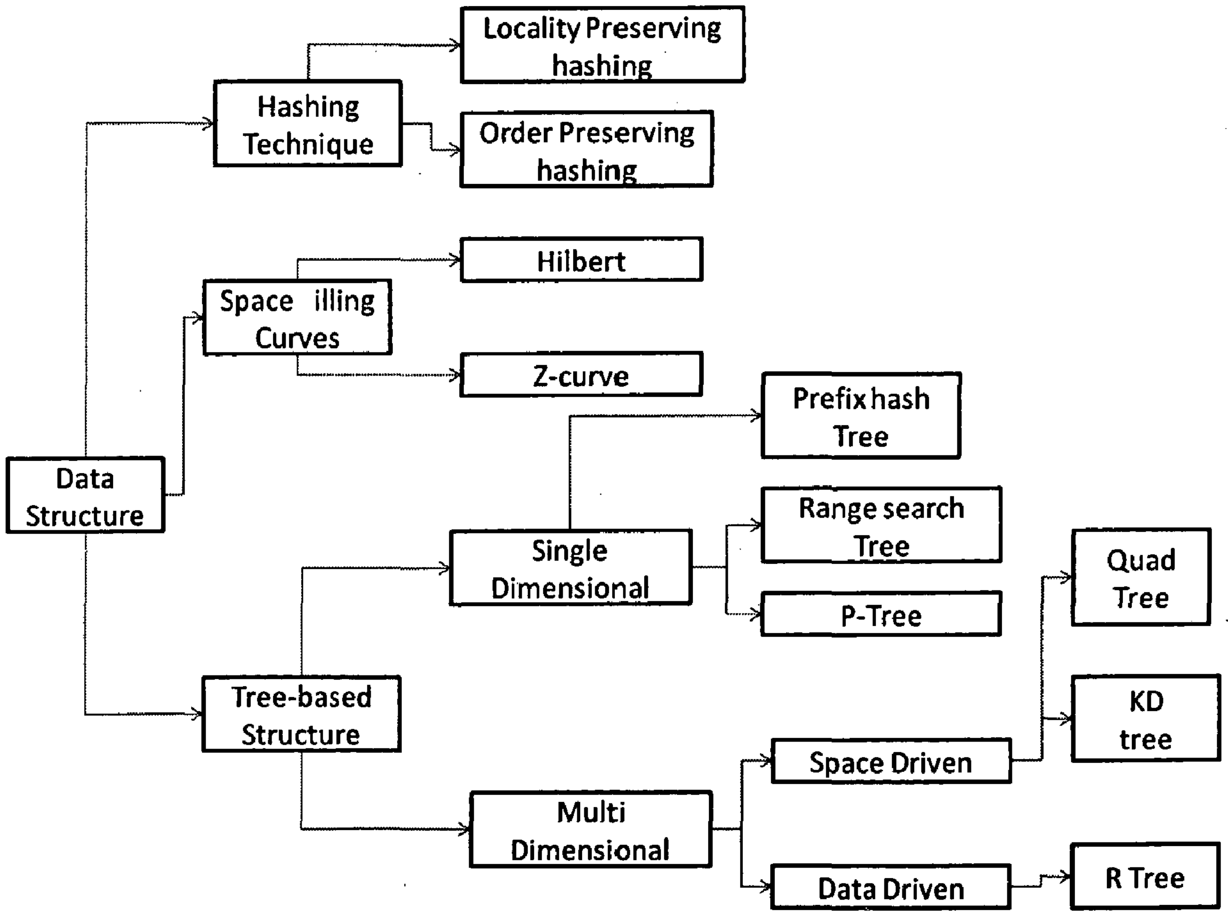


Figure 2.1 Peer to Peer Resource Discovery approaches (figure adapted from [6])

2.3.1 QuadTree: Using a Distributed QuadTree Index in Peer to Peer Network:

Tanin and Harwood [8] propose a distributed quadtree index that adopts an MX-CIF quadtree [21] for accessing spatial data or objects in P2P networks. A spatial object is an object with extents in a d-dimensional setting. A query that seeks all the objects contained in or overlapping a particular spatial region is called a spatial query. Such queries are resolved by recursively subdividing the underlying d dimensional space and then solving a possible simpler intersection problem. This recursive subdivision utilizes the basic quadtree representation.

2.3.2 MAAN: A Multi-Attribute Addressable Network for Grid Information Services:

Cai et al. [9] present a multi-attribute approach for enabling Grid Resource discovery. They extend the CHORD protocol to support range queries. MAAN address the d-dimensional range query problem by mapping the attribute values to the chord identifier space by a uniform locality preserving hashing. In this for every attribute dimension, a separate chord overlay is maintained. For attribute with numerical values, MAAN applies locality preserving hashing functions to assign an identifier to the CHORD m-bit identifier space. MAAN also supports Multi-attribute query resolution by extending the above single attribute range query routing algorithm. The system maintains a separate overlay/mapping function for every attribute.

2.3.3 HP-protocol

Andrejak and Xu [10] extend the CAN routing mechanism to support 1-dimensional range queries. This system uses the inverse SFC mapping from a one dimensional space to a d-dimensional space, to map a resource to peers based in a single attribute (for example, memory). It uses CAN as its overlay topology and the range of possible values for the 1 dimensional resource attribute is mapped onto CAN's d-dimensional Cartesian space. For each attribute/dimension a separate CAN space is required. To locate a resource based on multiple attributes, the proposed system iteratively queries for each attribute in different CAN space. Finally, the intersection of results for different attributes is taken.

2.3.4 Super-P2P R*-Tree:

Liu and Lee [11] extend the d-dimensional index R*-tree [21] for supporting range queries in a super peer based P2P system. The resulting distributed R*-Tree is referred to as an NRtree. Routing in the distributed d-dimensional space is accomplished through the CAN protocol. The d-dimensional distributed space is partitioned among super peer networks based on the minimum bounding Rectangle (MBR) of objects/points. Each partition (super peer network) refers to an index cluster (i.e. a MBR) and can be controlled by one or more super-peer. Effectively an index cluster includes a set of passive peers and super-peers. Every index cluster maps to a zone in the CAN based P2P space. The functionality of a super peer is similar to a router, it keep tracks of other index-clusters, performs inter-cluster routing, indexes data in other super peer partition and maintains cluster specific NR-tree. Every passive peer joins the network by contacting any available super peer. The contacted super peer routes the join request to the other super peer. Every passive peer maintains a part of the cluster specific NR tree.

The bulk of query processing load is coordinated by superpeers. Superpeers can forward query to its passive peers, in the case the indexed data is managed by them. Every Lookup request is forwarded to the local super-peer, if the requested indices are not available in the local zone. Peer initiating range query usually send the look up rectangle.

In case of a range query, the contacted super-peer routes the query to index cluster where the centroid of the query maps to. The owner of this index cluster is referred to as primary super-peer. The primary super peer searches its NR tree and finds passive peers with index intersecting the query region. The passive peers directly reply to the query initiating peer when a match occurs. Every lookup query has a TTL factor, which controls the life time for a query in the network.

2.3.5 Mercury

It supports multi-attribute based information search. Mercury [12] handles multi-attribute lookups by creating a separate routing hub for every resource dimension. Each routing hub represents a logical collection of nodes in the system and is responsible for maintaining range values for a particular dimension. Further each hub is part of circular overlay network.

2.3.6 SCRAP: Space Filling Curves with Range Partitioning

The work in [13] proposes two approaches for enabling multidimensional range queries over the CAN DHT. The d-dimensional data is indexed using the well known spatial data structures: Z-curves and KD-tree. First scheme is referred to as SCRAP: Space Filling Curves with Range Partitioning. SCRAP involves two fundamental steps: the d-dimensional data is first mapped to a 1-dimensional using the Z-curves and then 1-dimensional data is contiguously range partitioned across peers. Each peer is responsible for maintaining data in the contiguous range of values. Resolving multidimensional range queries in SCRAP network involves two basic steps: mapping multidimensional range queries into single dimensional range query using the SFCs, and routing the 1-dimensional range queries to the peers that indexes the desired look-up value. For routing query in 1-dimensional space the work proposes a scheme based on skip graph. A skip graph is a circular linked list of peers, which are organized in accordance with their partition boundaries. Additionally, peers can also maintain skip pointers for faster routing. Every peer maintains skip pointers to O(log(n)) other peers at an exponentially increasing distances from itself to the list. A single dimensional range query is resolved by the peer that indexes minimum value for the desired range. The message routing is done using the skip graph peer lists.

2.3.7 MURK: Multidimensional Rectangulation With KD Trees.

Other approach [13] referred to as d-dimensional Rectangulation with Kd-trees (MURK). In this scheme, d-dimensional space (for instance a 2d space) is represented as "rectangles" i.e., (hypercuboids in high dimensions), with each node maintaining one rectangle. In this case, these rectangles are used to construct a distributed KD-tree. The leaf nodes in the tree are stored by the peers in the network. Routing in the network is based on the following schemes:

• CAN DHT is used as basis for routing the multidimensional range query.

• Random pointers-each peer has to maintain skip pointers to random peers in the network.

This scheme provides similar query and routing efficiency as multiple realities in CAN; and space-filling skip graph-each peer maintain skip pointers to O(log(n)) other peers at exponentially increasing distances from itself in the network. Simulation results indicate that random and skip-graph based routing outperforms the standard CAN based routing for multidimensional range query.

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

The optimized information discovery system (OID) [14] is based on Space Filling Curves. This 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.

2.3.9 Distributed segment tree (DST)

Distributed segment tree (DST) [15] is a layered DHT structure that incorporates the segment tree concept, for the purpose of efficient support of range query. They introduced the segment tree concept and its properties as the basis of DST's range query. DST essentially tolerates more redundancies to achieve efficiency. It possesses parallelizability in query operations and can achieve O(1) complexity for moderate query ranges. Since DST is built on top of DHT, so it has all the advantages of DHT such as scalability, robustness etc.

2.3.10 Structured Segment Tree based

It [16] does not use the DHT logic, but embeds the structure of the segment tree into the P2P system. The topology of a Structured Segment Tree is actually the structure of a segment tree. It can fully reflect the properties of the original segment tree. Each peer in a Structured Segment Tree represents a node of a segment tree. Node intervals at the same level are continuous and will not overlap. The union of node intervals at a level with full nodes is the whole range which the P2P system can support. When searching a data range, the Structured Segment Tree method ends as many numbers of requests as needed. In addition sibling links are added to preserve the locality and boost the search efficiency.

2.4 Supporting Range Queries in P2P Systems

Supporting range queries in P2P systems is an important and challenging problem and has been addressed extensively. A variety of systems have been proposed and evaluated recently. Some of these systems allow multi-attribute range queries, while others only support a single attribute.

The efficiency of these systems depends on the way the data is mapped onto the nodes. Maintaining data locality is very important, since it is desired that minimum number of nodes are contacted for each range queries. But hashing used in DHT based systems destroys the data locality and can make range queries very expensive. The simplest approach is to use the exact values of the attribute. But this approach limits the search to single attribute ranges or imposes a more complex overlay network such as MERCURY where multiple simple overlays, one for each attribute, are mapped onto the same set of nodes.

Locality-preserving indexing schemes result in non uniform data distribution at nodes. As a result, explicit load balancing mechanism has to be employed. For example a node moves in a more crowded part of the overlay, leaving and rejoining the system.

Since DHTs and their hashing mechanisms do not preserve data locality, we use SFCs to build the index.

Spatial indices (QuadTree, KD-Tree, Hilbert/Z-curves, R-tree, R*-tree) are better suited to handling the complexity of grid resource queries than one dimensional data indices. Spatial indices are superior to one dimensional index as they incur fewer messages for multidimensional object lookups and updates. However, even spatial indices have routing load balance issues in case of a skewed data set. But, they are more scalable in terms of the number of hops and messages generated while searching in a multidimensional space.

Chapter 3

Peer to Peer Information Discovery Systems

3.1 Introduction

Information discovery is the important area where P2P technologies continue to be successfully applied. The key strengths of the P2P architecture such as the ability to utilize the peer resourced, self organization, fault tolerance and increased availability make it an attractive solution for this class of application.

P2P information discovery systems enable information sharing in a network of peers. The network of peers is typically an overlay network, defined at the application layer of the protocol stack (TCP/IP, ISO/OSI). A peer can have multiple roles. It can be a server (it store data and responds to query requests), a client (it request information from other peers), a router (it routes message based in its routing table), and a cache (it caches data to increase the system's availability).

The nodes in a typical P2P system self-organize into an overlay network at the application level. Search systems such as Gnutella use an unstructured overlay that is very easy to maintain. However, these systems do not scale well as they use a flooding based search mechanism, where each node forwards the query request to all or some of its neighbors. The systems built on unstructured overlays do not guarantee that all existing information that matches a query will be found. Structured overlay networks address these issues.

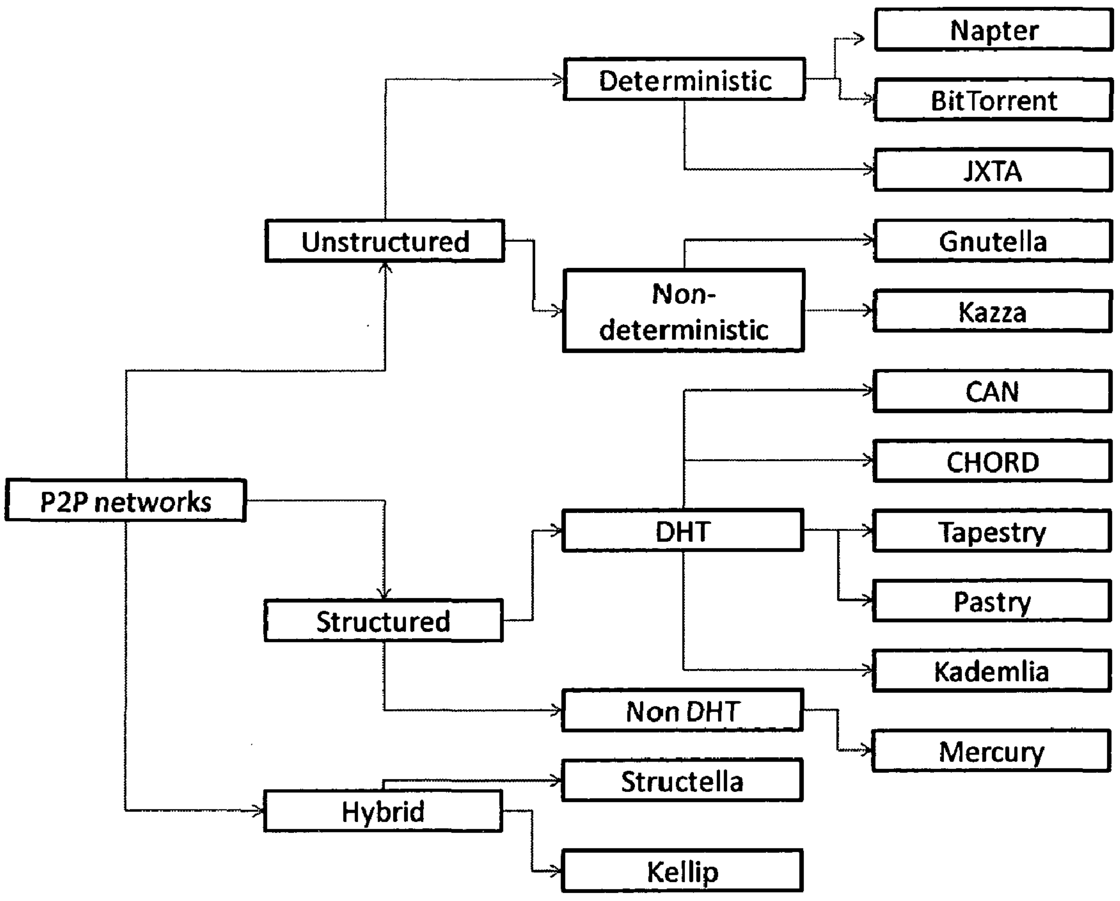


Figure 3.1 Classification of P2P systems (adapted from [6])

3.2 Structured Overlays of peers

In structured overlays, the topology of the peer network is tightly controlled. The nodes work together to maintain the structure of the overlay in spite of the dynamism of the system (i.e., nodes dynamically join, leave and fail). Maintaining the overlay structure can lead to some overheads, but data lookup in these systems is very efficient and can be guaranteed. for example CHORD system organizes nodes in a ring. The cost of maintaining the overlay when a node joins or leaves is O($\log^2$N) in this case, where N s the number of nodes in the system. In addition to this cost, each node periodically runs checks to verify the validity of its routing tables and repairs them if necessary. Similarly CAN organize nodes in a multidimensional toroidal space. Each node in CAN is associated with a hypercube region (zone) and has its neighbors the nodes that "own" the adjacent hypercubes. The cost of a node join is O(d) and a background algorithm that reassigns zones to nodes make sure that the structure of the overlay is maintained in case of node failures.

3.3 Distributed Hash Tables (DHTs) and DHT Routing

A hash table is a natural solution for a lookup protocol; given an identifier (a key) the corresponding data is fetched. Data stored in hash tables must be identified using unique numeric keys. A hashing function is used to convert the hash the data identifier into a numeric key. In P2P lookup system the hash table is distributed among peers, each peer storing a part of it. Consistent hashing is typically used to ensure a uniform distribution of load among the peers.

As DHTs build on structured overlays, the placement of data and the topology of the overlay network are highly structured and tightly controlled, allowing the system to provide access guarantees and bounds. Since the mapping of data elements to indices in the index space is based on unique data identifiers (i.e. filenames), only these identifiers can be used to retrieve the data. Complex queries such as range queries and keyword searches are not efficiently supported by these systems. Examples of data lookup systems include Chord [1], CAN [2], Pastry [3] and Tapestry. These systems have been used to construct persistent storage systems, cooperative distributed file systems, distributed naming systems, cooperative web caches and multicast systems.

The simplest DHT-based system operates like this. A user wants to publish a certain file in the network. The name of the file is hashed to produce unique file ID: the key. Then the function lookup(key) is called to estimate where the lookup pair(key, value), in this case (file ID, file location)-pair will be stored. When the appropriate node is found, this pair is propagated in DHT overlay to the calculated node ID. If another user wants to obtain this file the reverse procedure is initiated. First, he enters the file name, then via hashing its ID is obtained and fed into the lookup(FileID) function to get the ID of the node that stores the lookup pair. Via DHT-based routing the request for the File ID reaches the appropriate node ID and the file location is returned. Finally the user can download the needed file.

The main operation used in DHT-based systems is the lookup(key) function. It lies in the core of most other operations of the DHT-based system, such as a store operation (put(key, value)) or a retrieve value operation (value = get(key)).
The routing algorithm depends on the overlay used and its efficiency has a direct impact on the scalability of the system. Each node in a DHT based P2P lookup system has a unique numeric identifier which is a string of bits of a specified length.

Furthermore, each node in the system maintains a routing table containing pointers to a small number of other nodes. When a node receives a query for a key that is not stored locally, it routed the query to the node in its routing table that places the query nearest to the destination.

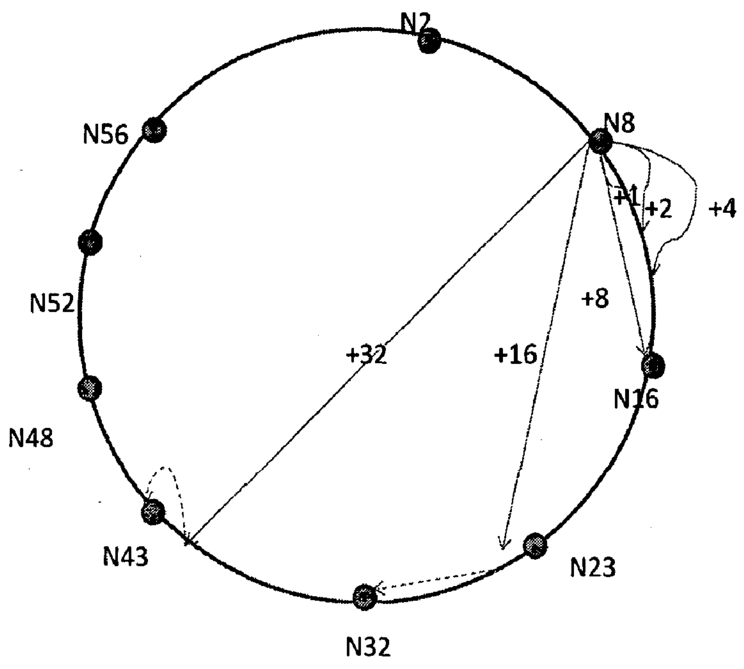
3.4 CHORD Overlay

Chord [1], developed by a group of researchers at MIT, is one of the first P2P overlay system based on distributed hashing table (DHT).

3.4.1 Address space: Chord uses the consistent hashing to assign each node and each key an m-bit identifier (Id), where m is a pre-defined system parameter. Ids fall into the range from 0 to 2^m-1. Nodes are ordered on an identifier circle modulo 2^m, as shown in Fig 3.1. A key is stored at its successor node, defined to be the next node in the identifier circle in the clockwise direction. The predecessor node to a node or key is the next node in the identifier circle in the counter-clockwise direction.

3.4.2 Routing table and key lookup: The routing table of Chord nodes consists of two parts. The first part includes a finger table with m entries, and the predecessor of this node. Assume the node Id is n. The i^{th} entry in the finger table points to the node whose Id is the closest to $n+2^i-1$ in the clock wise direction at identifier circle. The first entry in the finger table is the successor node of the current node. Predecessor node plus the finger table guarantees the correctness of key lookup service, as described below.
The second part of the routing table is a successor list of size r. In addition to the immediate successor node maintained in the finger table, other closest (r- 1) success nodes are also recorded. The successor list improves the robustness of Chord protocol, and

allows Chord to perform correctly in the face of peer churn, i.e., dynamic peer arrivals and departures.



Finger Table

N8+1	**N16**
N8+2	**N16**
N8+4	**N16**
N8+8	**N16**
N8+16	**N32**
N8+32	**N43**

Figure 3.2 CHORD Finger Table

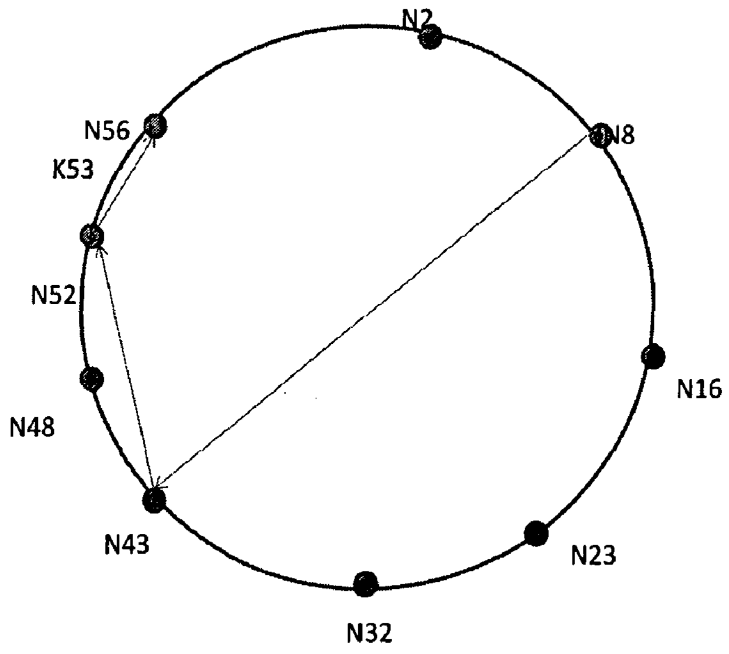


Figure 3.3 CHORD Key Lookup

A key lookup request is routed along the identifier. Upon receiving a lookup request, the node first checks if the lookup key Id falls between this node's Id and its successor's Id. If it does then it, returns the successor node as the destination node and terminates the lookup service. On the other hand, if the lookup key Id does not belong to the current node, the node relays the lookup request to the node in its finger table with Id closest to, but preceding, the lookup key Id. The relaying process proceeds recursively (or iteratively) until the destination node is found. A key lookup example is depicted in Fig. 3.2. This figure shows the finger table of Node 8 (N8). Node 16 (N16) appears in four entries in the finger table, while Node 32 (N32) and Node 43 (N43) are also in the finger table. Figure 3.3 depicts the stages for the lookup of key 53 starting from Node 8. It has been shown that the number of routing steps is at the order of O(logN), where N is the total number of Chord nodes.

3.4.3 Node join and leave: The newly arrived node in Chord first uses consistent hashing to generate its Id. It then contacts the bootstrapping node, the node already in the Chord, to lookup the successor of its Id. This successor node becomes new node's successor node. The new node uses the stabilization protocol, which is described below, to have a fully correct routing table.

Stabilization protocol is designed to maintain routing tables' correctness in the face of peer churns. It is executed periodically at the background of individual nodes. The stabilization protocol includes following two major functions: Stabilize( ): allows nodes to learn about newly joined nodes and to update their successor(s) and predecessor. Fix fingers( ): ensures finger tables are current and correct. Node failure/departure creates another challenge to the Chord protocol. The departure of a node leaves its predecessor node's successor pointer invalid, which could affect the routing correctness. To address this issue, Chord maintains a successor list of size r. The successor list can be stabilized using slightly changed stabilization protocol. It's proven that with size r =O(logN), where N is the total number of nodes in Chord, the lookup can still succeed with high probability even if every node fails with probability of 1/2.

Chapter 4

Space Filling Curves

4.1 Introduction

Space filling curves [24] are continuous functions which map multidimensional space into one dimension. It was originally formulated by Giuseppe Peano in 1890. It demonstrates that infinite number of points in a unit interval has the same cardinality as the infinite number of points in any bounded finite dimensional set. Since their invention, space filling curves have found application in a variety of fields, including mathematics, image processing, bandwidth reduction, cryptology, algorithms and scientific computing.

Since Peano introduced the first space filling curve many other curves have been constructed and studied. Among these families of curves generated by Hilbert find many applications. Hilbert curve impose an ordering on the points in finite square lattices. By considering the order on which the curve visits the points in an n dimensional lattice with 2^m points per dimension, we may assign an index to each point between 0 and 2^{mn} -1 . In context of database systems this enumeration is used to sort the points while preserving data locality.

The first graphical or geometric representation of the space filling curve is attributed to Hilbert. He illustrates the concept in 2 dimensional space and uses a binary radix to represent the coordinates of points in space.

4.2 Hilbert Curve

4.2.1 The Hilbert Curve in 2 Dimensions

Hilbert in 1891 illustrate the concept of a space filling curve in 2 dimensions by giving a method for the stepwise construction of an infinite sequence of finite curves, each of which is defined by linear ordering of the sub spaces resulting from the construction process. Hilbert then showed that this process defined in the limit a curve passing through all points in the space, this curve being everywhere continuous and nowhere differentiable.

Order of curve: The order of a curve is the number of steps or iterations for which the construction process has been carried out.

Approximation: A space filling curve of some finite order is said to be an approximation of a Hilbert space filling curve. It does not pass through every point in space but it passes through all of the centre points of a finite number of equal sized sub-square, the union of which comprises the whole of the unit square.

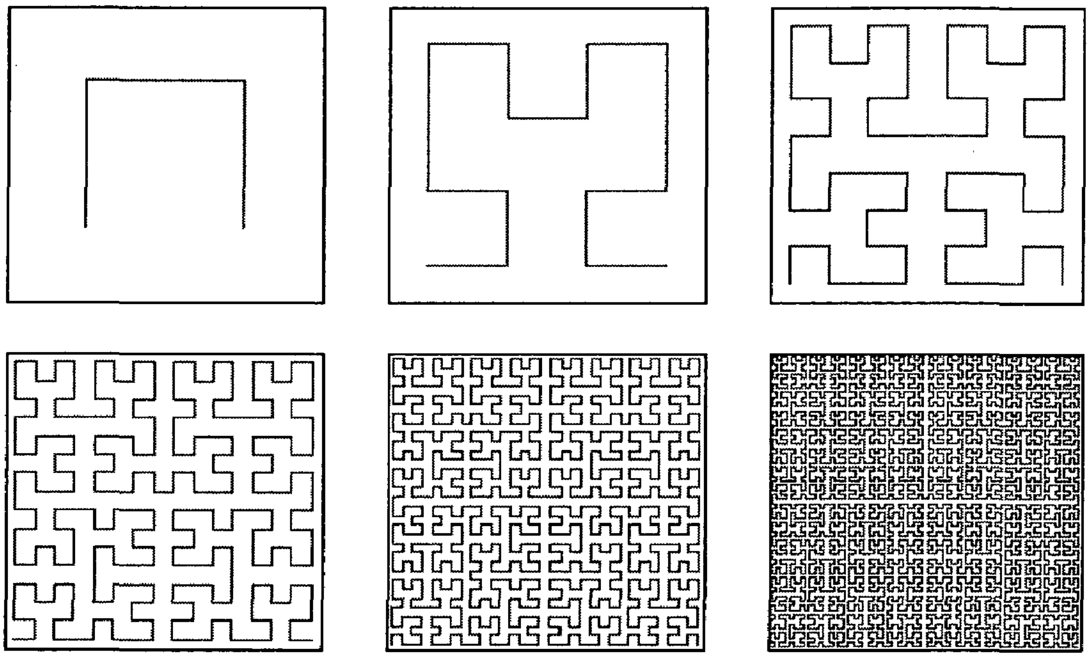


Figure 4.1 Hilbert Curve starting with 1st order from upper left to 6th order lower right corner

4.2.2 The Construction Process:

Step 1: As shown in figure 4.2 both the one Dimensional interval [0, 1] and the square [0, 1]^2 are initially divided into 4 congruent quarters. Each sub interval is then mapped to a different sub-square in such a way that sub-squares mapped to from adjacent sub intervals share a common edge.

The sub-squares are thus ordered. This is expressed graphically bellow by drawing a line made up of straight segments, passing through their centre points in the sequence in which they are mapped to from successive sub-intervals of line. The line passing though the centre points of sub-squares is referred to as a first order Hilbert curve.

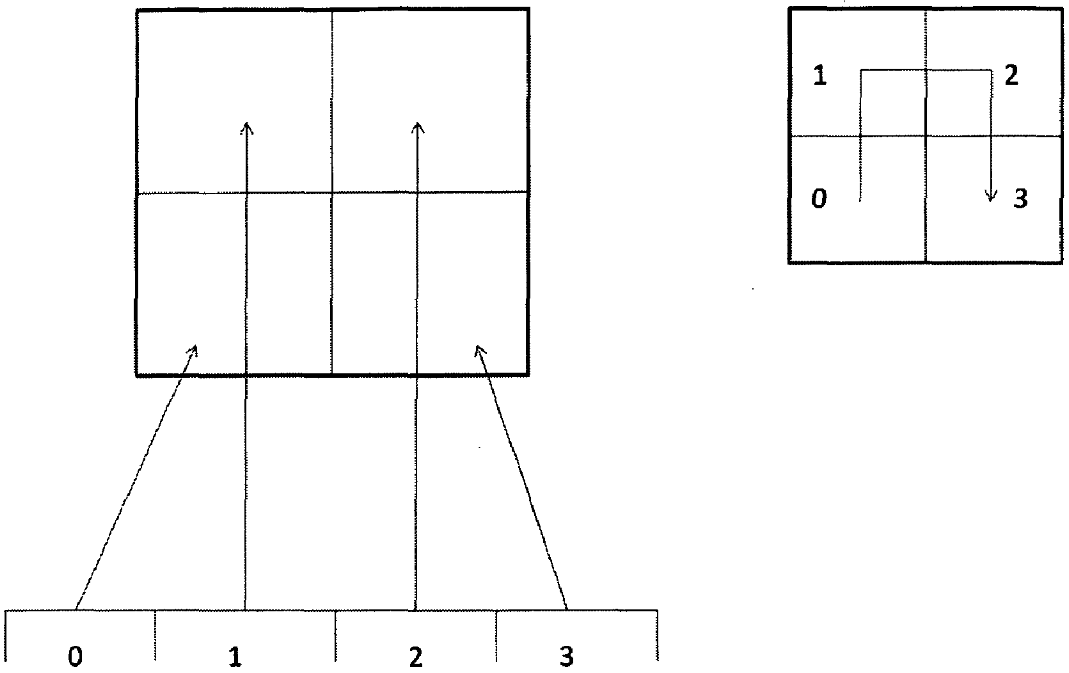


Figure 4.2 First Order Hilbert Curve: Mapping between Sub-squares and Sub-intervals in 2 dimensions

Step2: The Process of Division of intervals and squares is repeated for each of the four pairs of sub-intervals and sub-squares produced in the first step. This results in 4 groups of 4 equal sized sub intervals and sub-squares.

4.2.3 Higher Order Approximations

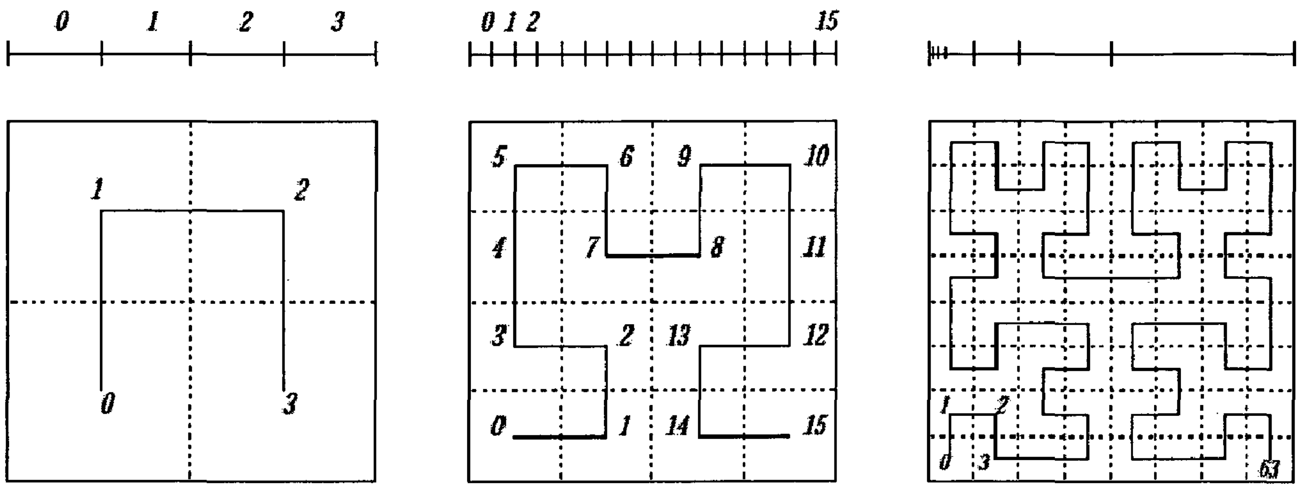
Given the order k curve defined on a $2^k * 2^k$ lattice, we may refine it to visit all points on a $2^{k+1} * 2^{k+1}$ lattice as follows.

(I) Place a copy of original curve rotated clockwise by 90 degree in the lower left sub grid.

(II) Place a copy of the original curve rotated in each of upper to grids.

(III) Place a copy of the original curve rotated counter clockwise by 90 degree in the right sub grid.

(IV) Connect these four disjoint curves in the obvious manner.



(a) 1st order (b) 2nd order (c) 3rd order

Figure 4.3 Hilbert curve and sequence number in 2 dimensions (Adapted from [17])

For example in the figure above (a) shows the first order Hilbert curve. Now this curve is rotated 90 degree clockwise and placed in lower left sub grid of (b).and copy of it is placed in two upper sub grid and a 90 degree anticlockwise rotated first order curve is placed in lower right sub grid. So we get second order curve. Similarly we can get 3rd order curve (c) in 2 dimension using 2nd order curve (b).

4.2.4 The Hilbert Curve in Higher Dimension

The concept of space filling curves as described by Hilbert can be extended in to higher dimensional space. For example in 3 dimensions, we begin by dividing a cube into 8 sub-cubes and order them so that sub-cubes which are mapped to from adjacent sub-intervals share a common face.

Hilbert Curves in higher dimensional space differ importantly from the 2-dimensional curve, because once an arbitrary first order curve has been drawn, there is a greater choice in ways of transforming the first two points of the curve. The existence of alternatives complicates the problem of expressing the Hilbert curve algorithmically.

4.2.5 Hilbert First order Curves in 3 Dimensions

Generally in n dimensions we note that first order Hilbert curve passed through 2^n hyper-cubes and that with each increment in the order of a curve, the granularity of the coordinate space for the centre points of these is increased by a magnitude of 2^n. Thus the number of points on a curve of order k is given by

$$2^{kn}$$

And so maximum one dimensional sequence number requires k*n binary digits to represent it. So number of digits required to represent the sequence number corresponding to a point equals to the number of digits required to represent the n coordinates of the point.

4.3 Compact Hilbert Indexes

Consider an n-dimensional data set consisting of points set P of precision $\{m_0, m_1, \ldots\ldots \ldots m_{n-1}\}$ in n dimensions. Storing a point of data require $M = (m_0 + m_1 + \ldots\ldots + m_{n-1})$ bits. However, a Hilbert index must be calculated with respect to a hypercube of precision $m = \max_i \{m_i\}$ and requires m*n bits of storage. In Grid attributes require variable precision, for example Price, OStype and FileSystemType may require precision of 9, 3 and 3 bits respectively. Points in their native space require 9+3+3 = 15 bits to store, while the associated Hilbert indices will require 3*9 = 27 bits, representing a data expansion factor of 27/15 = 1.8.

So Hamilton et al. [5] have introduces the indexing scheme that preserve ordering of Hilbert indices completely, but require only m bits to represent. A simple method to do this is to walk through all the points in P, calculate their Hilbert indices, and sort them based on these Hilbert indices. Then, assign to each point its rank as an index. This index has the same ordering as the Hilbert ordering over P and requires only M bits to represent. But in order to generate such index we must enumerate the entire space which has prohibitive cost. Efficient algorithm for generating Compact Hilbert Indexes is described in [5].

4.4 Z-Order

The Z-order or Morton-order first proposed in 1966 by G.M. Morton is a space filling curve which is often used in computer science. Due to its good locality preserving behavior it is used in data structures for mapping multidimensional data to one dimension. The Z-value of a point is simply calculated by interleaving the binary representations of its coordinate values. Once the data are sorted into this ordering, any one dimensional data structure can be used such as binary search tree, B-Trees, skip Lists etc.

For example, if all the bits of the coordinates of a two dimensional point lying on a curve of order k, are given as:

$P = \{ x_1\ x_2\ x_3 \ldots x_k, y_1\ y_2\ y_3 \ldots y_k \}$

Where x_1 and x_k are the most and least significant bits respectively in a coordinate value in dimension x, then the Z-order derived key Z, which results from bit-interleaving is:

$Z = x_1\ y_1\ x_2\ y_2\ x_3\ y_3 \ldots\ldots x_k\ y_k$

Figure 4.4 (a) below show the curve arrising from this process of bit interleaving.

Generally, in n dimensions, the coordinates of P can be experessed as:

$P =\{ P1, P2, \ldots, Pn \}$

Where each Pi is a coordinate in dimension i and i is in the range [1...n]. Each Pi is composed of k bits and so the coordinates of P can also be expressed as:

$P = \{P1_1\ P1_2 \ldots P1_k, P2_1 P2_2 \ldots\ldots P2_k, \ldots\ldots Pn_1 Pn_2 \ldots\ldots Pn_k\}$

Where Pi_j is a single bit and j is in the range [1 ... k]. Thus Pi_1 is the most significant bit of the coordinate in dimension i and Pi_k is the least significant bit in the same coordinate as:

$Z = P1_1\ P2_1 \ldots Pn_1\ P1_2P2_2 \ldots\ldots Pn_2 \ldots\ldots \ldots P1_kP2_k \ldots.Pn_k$

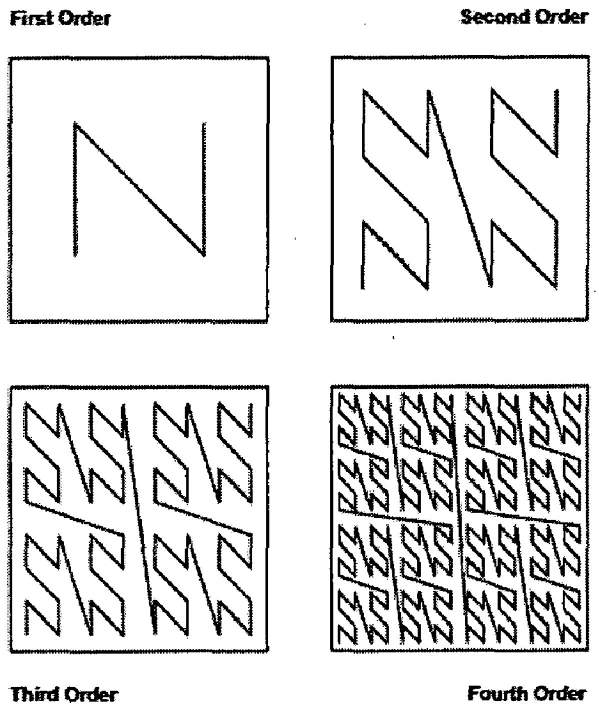


Fig a

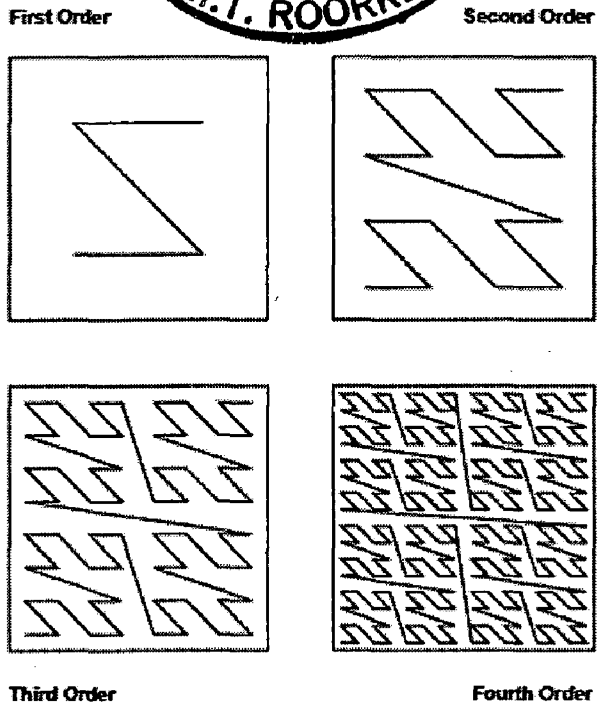


Fig b

Figure 4.4 Z -Curve from 1st to 4th order (figure adpated from [17])

Bits may be taken from the coordinated in a different order to produce derived key comprises k sequence of n bits.

$Z = y_1x_1y_2x_2y_3x_3 \ldots\ldots y_kx_k$

The Corresponding n dimensional Z-Order derived key of Point P given is expressed as

$Z = = Pn_1\ P(n\text{-}1)_1 \ldots.\ P1_1\ Pn_2\ P(n\text{-}1)_2 \ldots.\ P1_2 \ldots\ldots. Pn_k\ P(n\text{-}1)_k \ldots.\ P1_k$

Corresponding Z order for 2 dimension is given Figure 4.4 (b)

The process of bit interleaving results, importantly in a direct relationship between the values of bits in particular positions within particular coordinates and the value of bits in particular positions within derived-keys and vice versa. So Z-order algorithms provide the advantage of lower computational complexity over hilbert curve.

Chapter 5

Squid system

Squid system [4] consists of the following components:

1. A dimension-reducing, locality preserving indexing scheme
2. A structured overlay network
3. A mapping from indices to nodes in the overlay network
4. Load balancing mechanisms
5. A query engine for routing and resolving queries like range queries.

5.1 Mapping indices to Peers

Publishing data elements in Squid requires three steps:

1. Describe the data elements
2. Construct the index using the Hilbert SFC
3. Store the data (or a reference to the data) in the system using this index.

Figure 5.1 illustrates the publishing process.

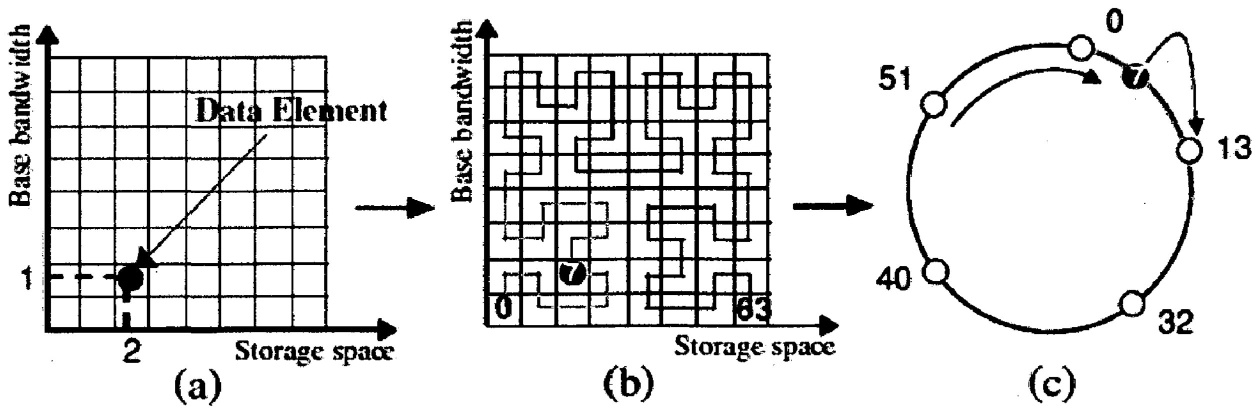


Figure 5.1 publishing the data element (Adapted from [4])

In this figure data element represent a computational resource (a machine with 2MB memory and 1 Mbps bandwidth) into the system

(a) The data element (2,1) is viewed as a point in a multidimensional space
(b) The data element is mapped to the index 7, using Hilbert SFC
(c) The data element is stored in the overlay (an overlay with 5 nodes and an identifier from 0 to 2^6-1) at node 13, the successor of index 7.

Each node is responsible for storing data elements associated with the segment between its predecessor and itself.

5.2 Query Engine

Processing a query consist of two steps (1) translating the query to relevant cluster of the SFC-based index space and (2) querying the appropriate nodes in the overlay network.

If the Query consists of exact match query it will be mapped to at most one point in the index space and the node containing the result is located using the overlay network's lookup protocol.

If the query is the range query, the query identifies a set of points in the keyword space that correspond to a set of points (indices) in the index space. In figure above the query (*, 4) identifies 8 data elements (the horizontal region). The index (curve) enters and exits the region three times, defining three segments of the curve or clusters. Similarly the query (4-7, 0-3) identifies 16 data elements, defining the square region in figure. The SFC enters and exits this region once, defining one cluster.

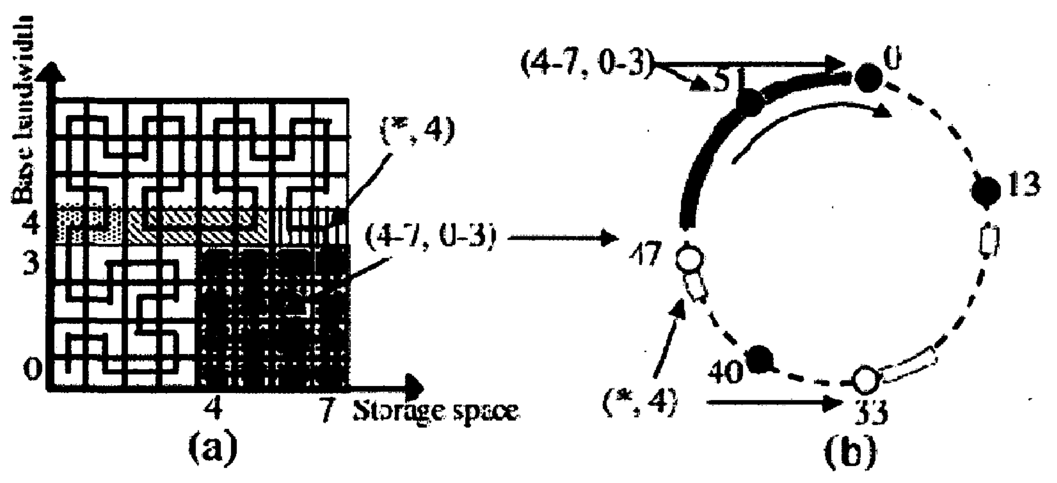


Figure 5.2 Searching in Squid (adapted from [4])

Each cluster may contain zero, one or more data elements that match the query. Depending on its size, a cluster may be mapped to one or more adjacent nodes in the overlay network. A node may also store more than one cluster. For example in figure node 33 stores 2 clusters and the 16-cell cluster is stored at nodes 51 and 0. Once the clusters associated with a query are identified, straightforward query processing would consist of sending a query message for each cluster. A query message for a cluster is routed to the appropriate node in the overlay network.

Distributed Query Processing in Squid System is described in detail in [4].

A query optimization is used to reduce the number of message involved. It is based on the observation that multiple sub-clusters of the same cluster may be mapped to the same node. To reduce the number of messages, each node sorts the sub-clusters in increasing order and performs a lookup for the sub-cluster with lowest identifier. Once the identifier of the node storing the cluster is known, it can be deduced that all the sub-clusters with identifier less than or equal with the node's identifier will also be stored at the node. A lookup for each sub-cluster is not required. All the sub-clusters can be aggregated and sent as a single message.

5.3 Balancing Load

In real world data elements are not typically uniformly distributed in the d-dimensional space. In certain regions there may be more data elements than others.

As the Hilbert/Z-order SFC-based mapping preserves locality, the index space will also preserve this locality. If nodes are uniformly distributed in the node identifier space, when the data elements are mapped to the nodes, load will not be balanced i.e. some nodes will get more data elements than others. Additional load balancing is required.

Load balancing at Node join: when a node join the system, the incoming node join the network which is most loaded. In this way, nodes will tend to follow the distribution of data from very beginning.

Load Balancing at runtime: At runtime load balancing between neighboring nodes can be done or other load balancing techniques can be used described in [4].

Chapter 6

Implementation Details

6.1 Tools used

Following Tools and Programming platform used in the implementation.

- Uzaygezen 0.1
- PlanetSim 3.0
- JDK 1.6
- NetBeans IDE 7.0
- Fedora Linux 12

6.1.1 Uzaygezen 0.1

Uzaygezen 0.1 [18] is a space filling curve library written in java with support for:

- Mapping from a multi-dimensional space into one dimension via the Compact Hilbert Index.
- The inverse mapping.
- Query building for databases with range query functionality, such as relational databases and more generally B-trees.

For compact Hilbert index and inverse calculation there are no limitations on the number of bits. For query building, the total precision must not exceed 62. In general it's a good idea to keep the precision of each dimension as small as possible.

We have used this library for calculation of Hilbert index from multi dimension and the range calculations.

6.1.2 Simulation Platform

Most popular simulators for Grid environment like SimGrid, GridSim does not provide the facility for Peer to Peer overlay network simulation.

There are many simulation tools [19] are available for simulation for peer to peer environment. These are NS-2 , PeerSim, P2PSim, OMNET++, OverSim, PlanetSim.

P2PSim is c++ based and support overlay like Chord, but its documentation is poor.

Oversim is open source c++ based network simulation framework for OMNet++ simulation Enviroment. It contains several models for structured and unstructured P2P protocols. OMNET++ has also been successfully used for peer to peer applications. Particularly, OMNET++ provides a rich environment that enables both packet-level simulations and high-level overlay protocols.
NS-2 is very popular in networking research, but it also does not provide simulation of P2P overlays. It has very low scalability.

All these network simulators are mainly aimed for packet-level protocols, and impose additional complexity to the user learning curve.

PlanetSim and PeerSim are java based Peer to Peer Simulators. They have high scalability and support some well known models and dynamic networks.

6.1.3 PlanetSim

PlanetSim [20] is an object oriented simulation framework for overlay network and services. In this Developers can work at two main levels: creating and testing new Overlay algorithms like Chord or creating and testing new services like DHT on top of existing overlays. PlanetSim also aims to enable a smooth transition from simulation code to experimentation code running in the internet.

Its API is very well designed so that source is quite readable and it has shorter learning curve as compared to other P2P simulators.

In PlanetSim Chord, Symphony and Skipnet Overlays are implemented and examples of Distributed Hash Table (DHT) functionality are given in it.

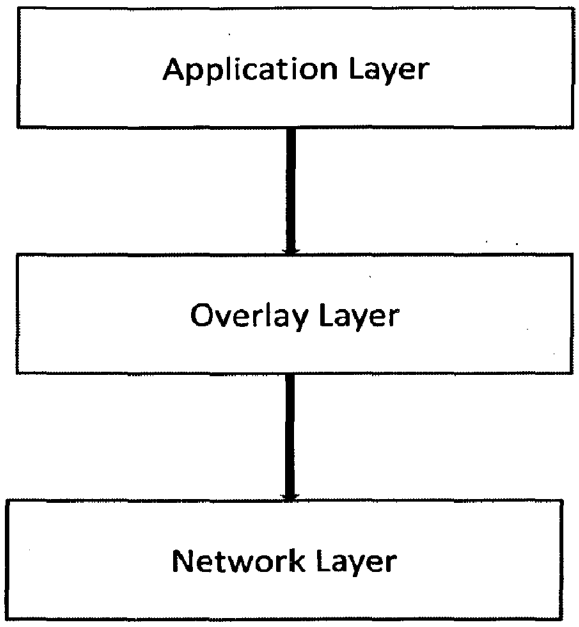


Figure 6.1 PlanetSim Layers

Application Layer: This layer is the facade to the underlying routing system of the simulator. This layer thus allows us to develop test application like DHT or Scribe multicast and Squid system (as we have done in the dissertation) over different overlays like Chord, Symphony or Skipnet.

Overlay Layer: At the overlay layer, the communication is bidirectional with both the application and network layers. By implementing the interfaces of this layer we can develop new overlay protocol like Chord or CAN.

Network Layer: An overlay can run on top of different underlying protocols. Developers can define their own networks with specific protocols. We are interested only in verification of SFC based mapping in the Squid system. So we are using simple network like 'CircularNetwork' provided in current version of PlanetSim. However it is feasible to use BRITE network [26] information to define more realistic networks.

6.2 Implementation Details

We implement the following classes for the application on the top of Chord Overlay Network

ChordApplication: it implement the Application interface of the simulator.

ChordMessage: It implement the Message interface of the simulator.

ChordTest: It extend the GenericApp class of the simulator.

6.2.1 Data generation

We have generated resources having 8 attribute randomly between their corresponding ranges as given in the table.

Resource Type	Range	No of Bits required
OSType	[0, 7]	3
FileSystemType	[0, 7]	3
Price	[0, 511]	9
Number of Available CPUs	[0, 64]	6
Maximum Clock Speed	[0, 15]	4
File system Available Space	[0, 127]	7
Free Virtual Memory	[0, 63]	6
Free Primary Memory	[0, 63]	6

Table 6.1 Resource description

For example OSType is assigned them to values between 0 to 7. Similarly other Attributes can be assigned.

NOTE: These Attributes are generated randomly between their corresponding ranges. This is not similar to real data. In the real data the number of resources between certain ranges will be high rather than uniformly distributed between full ranges. Also some attributes are interrelated such as Price of resource will be related to of other resources such as CPU clock-speed, memory.

So there will be load balancing issues on the real data. This dissertation work does not address this. Some load balancing techniques are provided in [4].

6.2.2 Data Placement

The SFC based mapping map the multidimensional data into 1 dimension. We have to place Key/Value to the peers in the overlay networks.

Chord Overlay is used to insert and lookup the Key/Value pair on the peers of overlay network. For a given key it can find the value corresponding to that key by querying over the overlay network.

Key: In our system is Hilbert/Z-order index derived from mapping multidimensional data to one dimension.

Value: It is the node/peer address at which the multidimensional resource from which key is calculated exists.

So when we query of entities having multiple resource like CPU, Memory, Bandwidth. Then we convert values of these resources into one dimension using Space Filling Curves described in chapter 4. For Query initiator this one dimensional value form the Key and it send the query using that key.

If we have to search the resources like CPU processing power, memory etc between some range then we take the lower and upper limits of all the attributes. We have to find the one dimensional keys (Hilbert/Z-Order) that exist between these lower and upper limits. These keys consist of many one dimensional ranges (clusters). We search the values corresponding to these keys using these clusters as described in chapter 5.

6.2.3 Cluster Calculation

For cluster calculations using Hilbert and compact Hilbert order we have used the Uzaygezen 0.1 [18]. For cluster calculation using Z-order we used the simple approach. Efficient Algorithm for Z-order Cluster (range) calculations is described in [27].

6.2.4 Data Scaling

In PlanetSim for storing in the 'key' must be at least 32 bit long and should be multiple of 32. The one dimensional keys generated from the SFC based mapping must be scaled so that they are 32 bit long i.e. in the [0, 2^{32}-1] range.

For example for 2 dimensional query including the attributed OSType and FileSystemType each has 3 bits for representation. So keys generated from SFC will be 6 bit (3+3) long. But Chord keys must be 32 bit, so these keys must be multiplied by 2^{32-6} or 67108864.

6.2.5 Query Algorithm:

We have developed the query algorithm which is different from Squid system. It is described below. The query is routed according to standard Chord overlay mechanism to responsible peer.

Query Processing at first node

1. Generate all the clusters in the query region
2. Sort the clusters.

   a = start point of first cluster ;

   b = end point of the first cluster;
3. Forward the query querying for the 'a'

6.2.5.1 Query Processing at the Chord Node

```
ID = current chord peer identifier
a= start point of next cluster
b= start end of next cluster
if(b < ID){
        do{
                add all the values between Keys 'a' and 'b' which this node store to the
                AnswerList
```

```
            a= start point of next cluster
            b= start end of next cluster
        }while(b<=ID);

        if (a< ID and b >= ID) THEN{
            add all the  values between Keys 'a' and 'b' which this node store to the
            AnswerList
            forward the query querying for the ID+1
        }else if(a> ID and b > ID){
            forward the query querying for the 'a'
        }
}else if((a < ID and b >= ID)){
        add all the  values between Keys 'a' and 'b' which this node store to the
        AnswerList
        forward the query querying for the ID+1
}
```

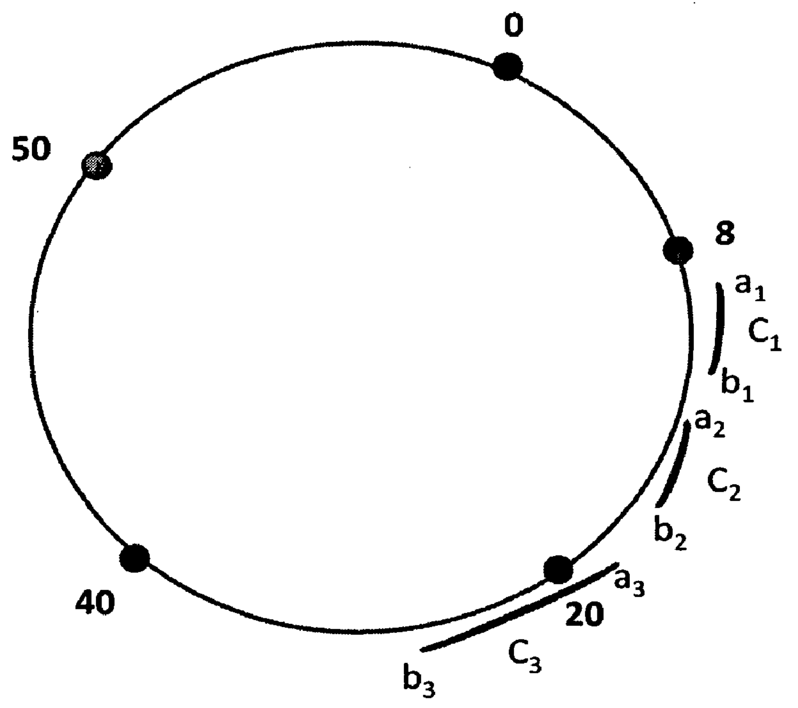


Figure: 6.2 Locations of clusters

We explain the query algorithm with the help of figure above.

In the figure above the Cluster List consist of Three Cluster C_1, C_2 and C_2 with Ranges [a_1, b_1], [a_2, b_2] and [a_3, b_3] respectively.

We query for the point a1 using standard chord routing algorithm. So the query arrives at node 20. So node 20 check the keys which are between range [a_1, b_1] and [a_2, b_2] and insert the values corresponding to those keys into the answer list.

For the cluster C_3 a_3<20(Node Identifier)<b_3. So value of Keys which are between a_3 and the node identifier 20 is inserted into the answer List And a query using key = 21(Node identifier)+1 is sent from node 20. So it arrives at the successor of node 20 i.e. at node 40 And Node 40 inserts the values of keys which are between 21 and b_3 into the AnswerList and send the message to the origin node.

The AnswerList will contain the identification/address of Entities having multidimensional resource.

6.2.5.2 Comparison with Schmidt Algorithm

Query processing algorithm given in [4] uses the recursive query processing and cluster generation. In this algorithm query message can arrive at those nodes which do not have the any data stored for the query. In Algorithm given above query message arrive only to those nodes which have data corresponding to the query.

But in our algorithm all the clusters must be identified at source node (the node which initiates the query) and these entire clusters have to be sent with the query message. In case number of clusters is high, message size will be larger. In the Algorithm given in [4] process of cluster generation is distributed over the network.

6.2.6 Range Query in Higher Dimension

The locality Preserving property of SFC begins to deteriorate for information spaces of dimensionality greater than five.

It is very time consuming to calculate the Hilbert indexes and the corresponding clusters for range query for 8-dimensions. And also corresponding number of clusters generated becomes very large.

Solution:

Earlier solutions have used the query over single attribute. For multidimensional range query they take the intersection of all the results from query over each of the single attribute.

But this solution has too much overhead. So this approach is not feasible.

So of querying over more than 6 dimensions we can divide the total attributes in the system into group to 4 attributes.

So number of SFCs mapping over overlay required = Total Number of Attributes / 4;

For example for query over the 8 attribute we have to use the 2 SFC based mapping.

1st mapping Attributes (OSType, FileSystemType, Price, Number of Available CPUs)

2nd mapping Attributes (Maximum Clock Speed, File system Available Space, Free Virtual Memory, Free Primary Memory)

We map each of these SFCs over the overlay network. And take the intersection of the results.

Chapter 7

Results and Discussion

7.1 Comparison between Hilbert Order and Z order

dimension	Query range	Number of Clusters Generated	
		Z Order	Hilbert order
2	[100 - 150] , [100 -150]	92	45
3	[100 - 150] , [100 -150], [100 - 150]	1515	2477
4	[100 - 110] , [100 -110], [100 - 110], [100 - 110]	2343	1177
4	[100 - 125] , [100 -125], [100 - 125], [100 - 125]	4214	2216
5	[100 - 110] , [100 -110], [100 - 110], [100 - 110], [100 - 110]	26336	14792
5	[100 - 125] , [100 -125], [100 - 125], [100 - 125], [100 - 125]	53866	30514
6	[100 - 110] , [100 -110], [100 - 110], [100 - 110], [100 - 110] , [100 -110]	292687	195757

Table 7.1 Clusters Generated

As we can see from the above data Hilbert SFC generate less number of clusters approximately 1.5 to 2 times of that Z Order SFC generate. But calculations of Hilbert indexes are complex and take more time as compared to Z order SFC.

7.2 Queries using Compact Hilbert Indexes

We generate the data as described in section 6.2.1.

Query Dimension	Attribute Range	Number of Clusters

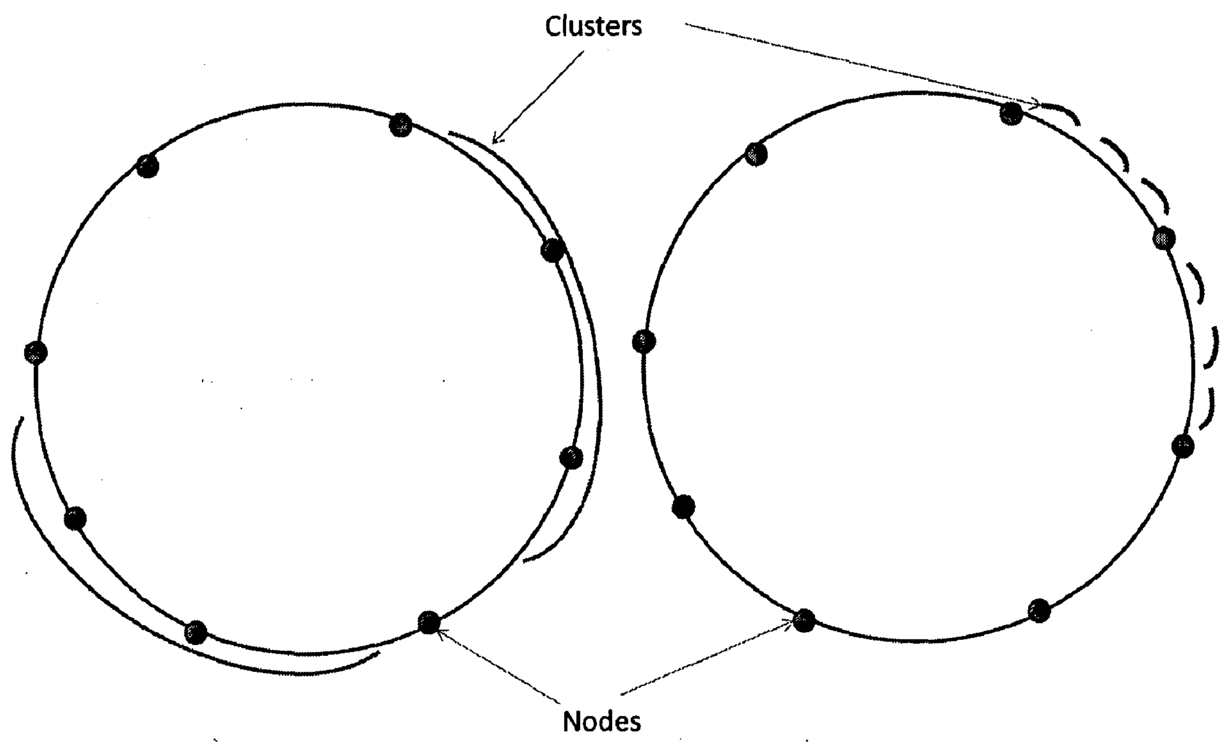
2	[2-4], [2-4]	3
4	[2-4], [2-4], [150-200], [35-40]	997
6	[2-4], [2-4], [100-200], [35-40], [5-10], [60-90]	329180

Number of Node	Number of Entities	Query Dimension	No. of Nodes Queried
100	100	2	14
100	100	4	6
100	100	6	9
100	1000	2	14
100	1000	4	6
100	1000	6	9
500	500	2	49
500	500	4	16
500	500	6	20
500	5000	2	49
500	5000	4	16
500	5000	6	20
1000	1000	2	96
1000	1000	4	30
1000	1000	6	31
1000	10000	2	96
1000	10000	4	30
1000	10000	6	31

Table 7.2 Query using Compact Hilbert SFC

We can observe that increasing the entities having multidimensional resource has no effect on the number of nodes queried.

We can see from the above table as number of dimension increases the number of nodes queried decrease. While earlier research on using space filling curves suggests that on increasing the dimensionality the locality becomes worse. This result can be explained on the observation that in lower dimension (i. e. 2 dimension) the size of d-dimensional space which we are searching is large. In querying using smaller dimension the number of cluster generated is small but sizes of clusters are large, So each cluster is mapped to many nodes and total number of nodes queried is more.



(a) Query over small dimension **(b) Query over large dimension**

Figure 7.1 query and cluster formation

This is explained in the figure above. In (a) only 2 clusters are generated and number of nodes queried are 6. In (b) 6 clusters are generated but number of nodes queried are 2.

In query over higher dimension (i.e. 6 dimensions) the size of d-dimensional space of range query become small because of specifying restriction on values in each dimension. So in querying using more dimensions the number of cluster generated becomes large but sizes of clusters are very small and numbers of clusters mapped to one node becomes large.

So although locality becomes worse in higher dimension but as size of query space decreases, the number of nodes queried becomes less when query over higher dimension.

7.3 8-Dimension query (using Compact Hilbert indexes)

Single SFC mapping using 8 attributes is built. Query over using the 8 attributes take much time for calculation of clusters which becomes too large in case of 8 dimensions. In our query it takes more than 30 minutes to calculate the clusters. As number of clusters becomes very high so query message size will become quite large if we send them in the single query message.

As explained in previous section eight dimension query may be resolved using 2 SFCs (we have used compact Hilbert indexes) each using 4 attributes. And taking the intersection of two results set.

Query Dimension	Attribute Range	Number of Clusters
4 (1st part)	[2-4], [2-4], [100-200], [20-45]	4027
4 (2nd part)	[2-12], [50-100], [20-40], [40-60]	12300

Number of Node	Number of data Entities	No. of Nodes Queried	
		1st part	2nd part
100	100	20	17

100	1000	20	17
500	500	59	60
500	5000	59	60
1000	1000	107	102
1000	10000	107	102

Table 7.3 8-dimensional query

7.4 Comparison of Z order, Hilbert and Compact Hilbert Indexes

For comparison we generate the data in range as described in section 6.2.1, but in 3rd dimension range is [0, 255] instead of [0, 511] so it takes 8 bits to represent.

Query Dimension	Attribute Range	Number of Clusters	
		Z-order	Hilbert Order and Compact Hilbert Order
2	[2-4], [2-4]	5	3
4	[2-4], [2-4], [150-200], [35-45]	2405	1150

Number of Node	Number of data Entities	Query Dimension	No. of Nodes Queried		
			Z-order	Hilbert Order	Compact Hilbert
100	100	2	11	14	13
100	100	4	3	3	9
100	1000	2	11	14	13
100	1000	4	3	3	9
500	500	2	37	49	51
500	500	4	5	3	34
500	5000	2	37	49	51
500	5000	4	5	3	32

1000	1000	2	68	96	99
1000	1000	4	5	3	62
1000	10000	2	68	96	99
1000	10000	4	5	3	62

Table 7.4 Comparison of Z order, Hilbert and Compact Hilbert Indexes

We can observe that in query in 4th dimension using Hilbert and Z-order SFC takes much less nodes to be queried. On analyzing the results we find that using Hilbert and Z-order data distribution is very skew i.e. most key/value pairs are stored at few nodes and many nodes don't store any data. Using compact Hilbert indexes data (key/value pairs) are more uniformly stored among the nodes, So compact Hilbert indexes are more suitable if the precision of dimensions are unequal not only because they use less bits to represent the key but also because they have better load balancing properties.

7.5 Dynamic Attributes

In grid computing many resources are dynamic i.e their values change with time. This can be incorporated in the system. If value of some resource changed then we have to search the node where key/value pairs are stored where key is the Hilbert/Z-order value of the computed from all the attributes and value is the address of the resource. We delete that key/value pair from that node. And insert the new key calculated from Hilbert/Z-order the current attribute value being address of the resource.

7.6 Peer failure /departure

In case of peer failure/ departure can be handled using standard mechanism that Chord [1] use and briefly described in section 3.5.

Chapter 8

Conclusions and Scope for future work

8.1 Conclusions

In this dissertation work multidimensional range queries over Chord overlay network is evaluated. We have used Space Filling curve to map multidimensional data to into dimension and then query for this data. We have used compact Hilbert indexes in addition of Hilbert and Z-order curve. The following conclusions can be drawn from this dissertation work.

- Numbers of clusters generated by using Z-order SFC are approximately 1.5 to 2 times from that generated using Hilbert SFC.
- For more than 6 dimensions calculation of Hilbert SFC clusters (ranges) becomes very time consuming. Z-order SFC has faster calculation time but numbers of clusters generated are very high in higher dimension. So it is not feasible to use SFC more than 6 dimensions.
- Although Number of Clusters increases with the Dimensions the number of nodes queried for higher dimension decreased. But it is known fact that locality properties of SFC become worse in higher dimension. This is due to query space over which searching is involved decreases by adding more dimensions.
- For query over higher dimension it is feasible to use the dividing the number of attributed in group of 4 attributes and use the separate SFC for each of these group and query over them and finally take the intersection of results.
- Using compact Hilbert indexes are more suitable if the precision of dimensions are unequal not only because they use less bits to represent the key but also because they have better load balancing properties than Hilbert and Z-order SFC.

8.2 Scope for Future Work

Squid system evaluated in this dissertation is used in real world. This system can be further evaluated or extended in terms of load balancing, peer failures and exploiting peer heterogeneity and some optimization in query engine and using cache for system availability.

Squid system is evaluated for Chord overlay network; in future it can be evaluated for different overlay networks like Skipnet, CAN etc.

Multidimensional range queries over P2P network is an important research topic. In this field as many researchers have proposed different kind of approaches. In future research these approaches can be evaluated or extended.

REFERENCES

[1] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet applications" *In Proceedings of the ACM SIGCOMM' 01*, pp. 149–160, San Diego, CA, August 2001.

[2] S. Ratnasamy et al., "A Scalable Content-Addressable Network" *In Proceedings of ACM SIGCOMM*, ACM Press, pp. 161-172, 2001,.

[3] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," *Proceedings of International Conference of Distributed Systems Platforms (Middleware)*, ACM Press, pp. 329-350, 2001,.

[4] C. Schmidt, M. Parashar, "Squid: Enabling search in DHT-based systems", *journal of Parallel and Distributed Computing* , Volume 68 Issue 7, Elsevier, pp 962–975, July 2008.

[5] Chris H. Hamilton, Andrew Rau-Chaplin, "Compact Hilbert Indices for Multi-Dimensional Data," *First International Conference on Complex, Intelligent and Software Intensive Systems* (CISIS'07), pp.139-146, 2007.

[6] R.Ranjan, A. Harwood and R. Buyya "Peer-to-peer-based resource discovery in global grids: a tutorial", *Communications Surveys and Tutorials, IEEE*, Vol. 10, No. 2, pp.6-33, 2008.

[7] E. Meshkova, J. Riihijarvi, M.Petrova, P. Mahonen, "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks," *Computer Networks Volume 52, Issue 11, Elsevier*, pp 2097-2128, 8 August 2008.

[8] E. Tanin, A. Harwood, and H. Samet. "A Distributed Quadtree Index for Peer toPeer Settings", *Proceedings of International Conference Data Eng*, pp. 254–55, 2005.

[9] Min Cai , Martin Frank , Jinbo Chen , Pedro Szekely, "Maan: A Multi-Attribute Addressable Network for Grid Information Services", *Proceedings 4th IEEE/ACM International Workshop. Grid Computing*, pp. 184–91, 2003.

[10] A. Andrzejak and Z. Xu "Scalable, efficient range queries for grid information services", *In Proceedings of the Second IEEE International Conference on Peer-to-Peer Computing* (P2P2002), pages 33-40, Sweden, Sept. 2002.

[11] B. Liu, W. Lee, and D. L. Lee, "Supporting Complex Multi-Dimensional Queries in P2P

Systems", *Proceedings 25th IEEE International Conference of Distributed Computing System*, pp. 155-64,Columbus, OH, 2005.

[12]A. R. Bharambe, M. Agrawal, and S. Seshan, "Mercury: Supporting Scalable Multi-Attribute Range Queries", *SIGCOMM '04*, pp. 353–66, Portland, OR, 2004,.

[13]P. Ganesan, B. Yang, and H. Garcia-Molina, "One torus to rule them all: Multidimensional queries in P2P systems", *In Proceedings of the ACM SIGMOD'04, WebDB Workshop*, pages 19–24, Paris, France, June 2004.

[14]Memon, D. Tiebler, F. Dürr, K. Rothermel, M. Tomsu and P. Domschitz, "OID: Optimized Information Discovery using Space Filling Curves in P2P Overlay Networks", In *Proceedings of 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08), pp. 311-319,* Melbourne, Australia, December 2008.

[15] C. Zheng, G. Shen, S. Li, and S. Shenker, "Distributed Segment Tree: Support of Range Query and Cover Query over DHT", *in proceedings of the 5th International workshop on peer to peer systems*, pp. 5.15 , Feb 2006.

[16] Y.I. Chang, C.C. Wu_, J.H. Sheny and T.L. Huang "Range Queries Based on a Structured Segment Tree in P2P Systems" in *17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems,* 2010.

[17] J.K. Lawder and P. J. H. King. "Using Space-Filling Curves for Multi-dimensional Indexing" In Advances in Databases, *17th British National Conference on Databases (BNCOD 17)*, vol 1832, Lecture Notes in Computer Science, pp. 20-35, July 2000.

[18] uzaygezen [Online] Available: http://code.google.com/p/uzaygezen/

[19]R. Bhardwaj, A. K. Upadhyay, V.S. Dixit, "An Overview on Tools for Peer to Peer Network" *Simulation International Journal of Computer Applications* (0975 - 8887) Volume 1 No. 19, 2010 .

[20] PlanetSim [Online] Available: http://projects-deim.urv.cat/trac/planetsim/

[21] Hanan Samet, "Foundations of Multidimensional and Metric Data Structures", Morgan Kaufmann 2006, pp.38-41, 289-295.

[22] Xuemin Shen, Heather Yu, John Buford, Mursalin Akon, "Handbook of Peer-to-Peer Networking", Springer-Verlag, 1st Edition 2009, , part III section 3.1 pp. 231-32.

[23] Doug Moore, "Fast Hilbert Curve Generation, Sorting, and Range Queries" [Online] Available: http://www.tiac.net/~sw/2008/10/Hilbert/moore/index.html

[24] Hans Sagan, "Space-Filling Curves", Springer -Verlag, 1 edition, September 2, 1994

[25] A. R. Butz, "Alternative algorithm for hilbert's space-filling curve," *IEEE Trans. On Computers*, vol. C, no. 2, pp. 424–426, 1971.

[26] BRITE: "Boston University Representative Internet Topology Generator" [Online] Available: http://www.cs.bu.edu/brite/

[27] H. Tropf and H. Herzog, "Multidimensional range search in dynamically balanced trees," *Angewandte Informatik*, vol. 23, no. 2, pp. 71-77, Feb. 1981.