OVERVIEW OF SPACE-FILLING CURVES AND THEIR APPLICATIONS IN SCHEDULING

Mir Ashfaque Ali[1] and S. A. Ladhake[2]

[1]Head, Department of Information Technology, Govt. Polytechnic, Amravati (MH), India.
[2]Principal, Sipna's College of Engineering & Technology, Amravati (MH), India.

ABSTRACT

Space-filling Curves (SFCs) have been extensively used as a mapping from the multi-dimensional space into the one-dimensional space. A space-filling curve (SFC) maps the multi-dimensional space into the one-dimensional space. Mapping the multi-dimensional space into one-dimensional domain plays an important role in every application that involves multidimensional data. Modules that are commonly used in multi-dimensional applications include searching, scheduling, spatial access methods, indexing and clustering. Space-filling curves are adopted to define a linear order for sorting and scheduling objects that lie in the multi-dimensional space. Space filling curves as the basis for scheduling has numerous advantages, scalability in terms of the number of scheduling parameters, ease of code development and maintenance. This paper elaborates the space-filling curves and their applicability in scheduling, especially in transaction and disk scheduling in advanced databases.

KEYWORDS

Scheduling, Space-filling Curve, Real-time Database, Disk Scheduling, Transaction Scheduling.

I. INTRODUCTION

Many people have devoted their efforts to find a solution to the problem of efficiently scheduling task or transaction with multi-dimensional data. This problem has gained attention in the last years with the emergence of advanced database system and operating system such as real-time databases, real-time operating system which need to schedule and process the task or transaction in an efficient way. Hence, techniques that aim to reduce the dimensionality of the data usually have better performance. One such way of doing this is to use a space-filling curve. A space-filling curve can transform the higher dimensional data into a lower dimensional data using some mapping scheme.

Space-filling Curves (SFCs) have been extensively used as a mapping from the multi-dimensional space into the one-dimensional space. A space-filling curve (SFC) [1] maps the multi-dimensional space into the one-dimensional space. Mapping the multi-dimensional space into one-dimensional domain plays an important role in every application that involves multidimensional data. Multimedia databases, geographical information systems, QoS routing, image processing, parallel computing, data mapping, circuit design, cryptology and graphics are examples of multi-dimensional applications. Modules that are commonly used in multi-dimensional applications include searching, scheduling, spatial access methods, indexing and clustering [2, 3, 4].

A space-filling curve is a way if mapping the multi-dimensional space into the one-dimensional space. An SFC acts like a thread that passes through every cell element (or pixel) in the multi-dimensional space so that every cell is visited exactly once. Thus, space-filling curves are adopted to define a linear order for sorting and scheduling objects that lie in the multi-dimensional space. Figure 1 gives examples of six two-dimensional space-filling curves. Using space-filling curves as the basis for scheduling has numerous advantages, like:

- Scalability in terms of the number of scheduling parameters,
- Ease of code development and maintenance,
- The ability to analyze the quality of the schedules generated, and
- The ability to automate the scheduler development process in a way similar to automatic generation of programming language compilers.

Mapping from the multi-dimensional space into the one-dimensional domain provides a pre-processing step for multi-dimensional applications. The pre-processing step takes the multi-dimensional data as input and outputs the same set of data represented in the one-dimensional domain. The idea is to keep the existing algorithms and data structure independent of the dimensionality of data. The objective of the mapping is to represent a point from the D-dimensional space by a single integer value that reflects the various dimensions of the original space.

The rest of the paper is organized as follows. Section 2 surveys some of the related work on space-filling curves. Section 3 describes about mapping in space-filling curves. Section 4 describes about space filling curves application in scheduling transaction in active and real-time database. Section 5 describes again its usage in disk request scheduling in multimedia databases. Finally we conclude in section 5.

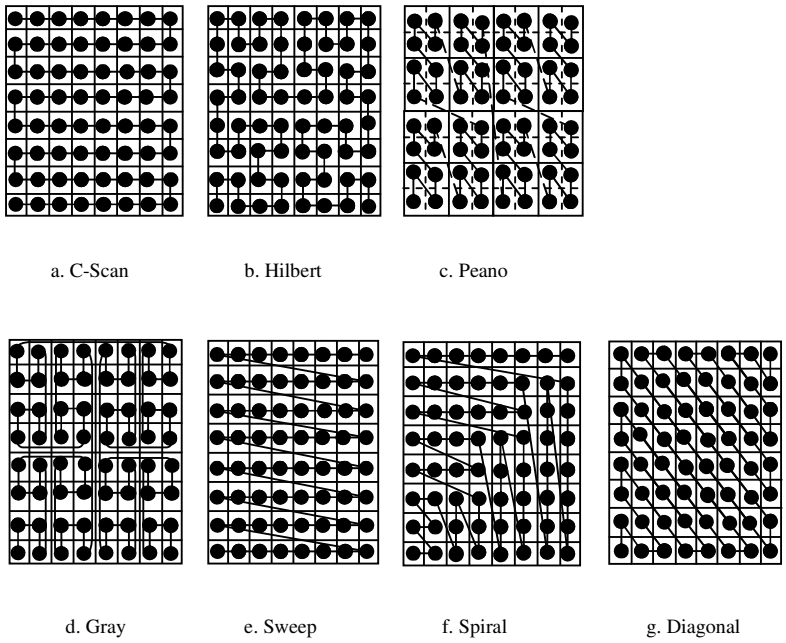

Figure1. Space-Filling curves examples.

II. RELATED WORK

The notion of space-filling curves has origins in the development (in 1883) of the concept of the Cantor set. Peano in 1890 and Hilbert in 1891 provided explicit descriptions of such curves. In 1890 Peano discovered a densely self-intersecting curve that passes through every point of the unit square. Purpose was to construct a continuous mapping from the unit interval onto the unit square. Peano was motivated by Georg Cantor's earlier counterintuitive result that the infinite number of points in a unit interval is the same cardinality as the infinite number of points in any finite-dimensional manifold, such as the unit square. The problem Peano solved was whether such a mapping could be continuous, i.e., a curve that fills a space [4].

Bokhari & Aref [5] apply 2D and 3D Hilbert curves to binary dissection of nonuniform domains while taking into account shape, area, perimeter, or aspect ratio of regions. Ou et al. [6] propose a partitioning based on SFCs that is scalable, proximity improving and communication minimizing. Aluru and Sevilgen [7] discuss load balancing using SFCs. They show how nonuniform and dynamically varying data grids can be mapped onto SFCs, which can then be partitioned over processors. Chatterjee et al. [8] show the applications of Hilbert curves to matrix multiplication. Recent research by Zhu and Hu [9] also describes the use of Hilbert curves for load balancing. In [10], Jagadish presents an analysis of the Hilbert curve for representing two-dimensional space. Moon et al. [11] analyze the clustering properties of the Hilbert curve and compare the performance of Hilbert curves with Z-curves. This paper also includes a good historical survey.

III. MAPPING IN SPACE FILLING CURVES

A space-filling curve must be everywhere self-intersecting in the technical sense that the curve is not injective. If a curve is not injective, then one can find two "subcurves" of the curve, each obtained by considering the images of two disjoint segments from the curve's domain. The two subcurves intersect if the intersection of the two images is non-empty. One might be tempted to think that the meaning of "curves intersecting" is that they necessarily cross each other, like the intersection point of two non-parallel lines, from one side to the other. But two curves (or two subcurves of one curve) may contact one another without crossing, as, for example, a line tangent to a circle does.

In general, space-filling curves start with a basic path on a k-dimensional square grid of side 2. The path visits every point in the grid exactly once without crossing itself. It has two free ends, which may be joined with other paths. The basic curve is said to be of order 1. To derive a curve of order i, each vertex of the basic curve is replaced by the curve of order i, which may be appropriately rotated and/or reflected to fit the new curve [5].

The basic Peano curve for a 2*2 grid, denoted N1, is shown in Figure 2. To derive higher orders of the Peano curve, replace each vertex of the basic curve with the previous order curve. Figure 2 also shows the Peano curve of order 2 and 3.

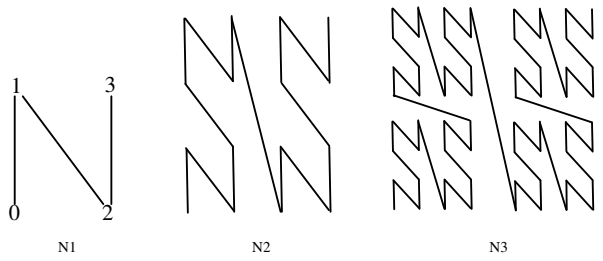

Figure 2. Peano curves of order 1, 2 and 3.

The basic reflected binary gray-code curve of a 2*2 grid denoted R1 is shown in figure 3 (a). The procedure to derive higher orders of this curve is to reflect the previous order curve over the x-axis and then over the y-axis. Figure 3 (a) also shows the reflected binary gray-code curve of order 2 and 3.

The basic Hilbert curve of a 2*2 grid, denoted H1, is shown in figure 3 (b). The procedure to derive higher orders of the Hilbert curve is to rotate and reflect the curve at vertex 0 and at vertex 3. The curve can keep growing recursively by following the same rotation and reflection pattern at each vertex of the basic curve [Lu, 2003]. Figure 3 (b) also shows the Hilbert curves of order 2 and 3. An algorithm to draw this curve is described in [Griffiths, 1986].

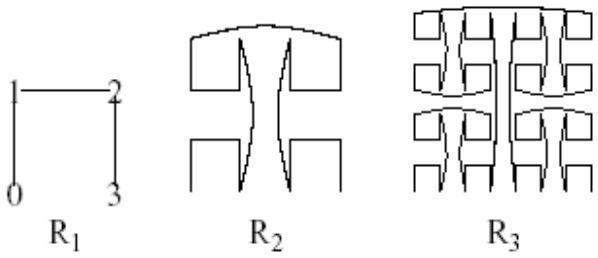

Figure 3 (a). Reflected binary gray-code curves of order 1, 2 and 3.

The path of a space-filling curve imposes a linear ordering, which may be calculated by starting at one end of the curve and following the path to the other end. [Orenstein & Merrett, 1984] used the term **z-**ordering to refer to the ordering of the Peano curve.
The Space-filling curves are used for scalability, fairness and intentional bias [Mokbel & Aref, 2001]. The SFC are scalable, when any new parameter comes into picture a new dimension can be added or number of points per dimension can be increased. The space-filling curve is said to be fair if it results in similar irregularity for all dimensions. The notion of irregularity is the measure of goodness for the mapping of each space-filling curve.

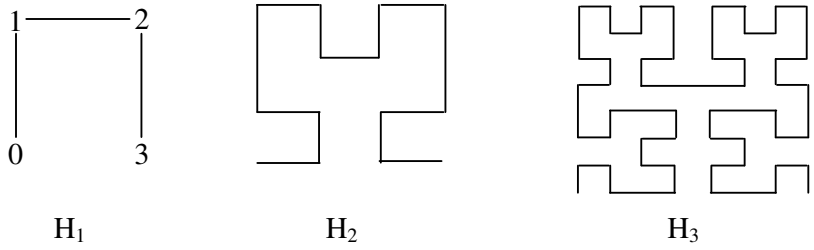

Figure 3 (b). Hilbert curves of order 1, 2 and 3.

IV. SCHEDULING TRANSACTIONS USING SFC IN DATABASES

In [12] a new transaction-scheduling scheme is proposed for real-time database system based on three-dimensional design by integrating the characteristics of value, deadline and criticalness. Here space-filling curves can be used as they are adopted to define linear order for sorting or scheduling. The space filing curves unnaturally considers value, deadline and criticalness information and gives a scheduling sequence. A CPU request is modeled by multiple parameters, (e.g., the real-time deadline, the criticalness, the priority, etc.) and represented as a point in the multi-dimensional space where each parameter corresponds to one dimension. Using a space-filling curve, the multi-dimensional CPU request is converted to a one-dimensional value.
A CPU request *T* takes a position in the thread path according to its space-filling curve value. They are then stored in the priority queue q according to their position in the thread path. The CPU scheduler walks through the thread path by serving all CPU requests in queue according to their path position, which is their one-dimensional value with a lower value indicating a higher priority. Figure 4 gives an illustration of an SFC-based CPU scheduler.

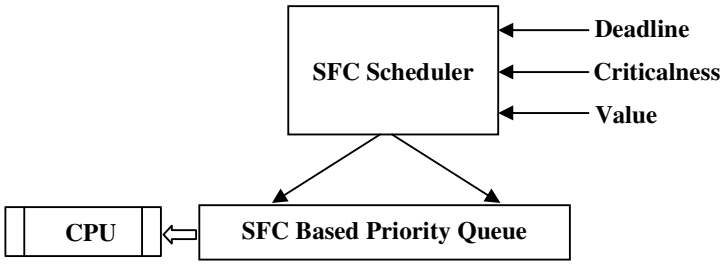

Figure 4. Space filling curve based CPU scheduler.

The space filling curves converts 3-dimensional space using the idea of bit interleaving which is used and described in [3, 5]. Every point in the space takes a binary number that results from interleaving bits of the two dimensions. The bits are interleaved according to the interleaving vector (0,1,0,1). This corresponds to taking the first and third bits from dimension 0 (x) and taking the second and fourth bits from dimension 1(y). The result of applying this bit interleaving is shown Table 1.
The sequence of few transactions obtained after mapping from 3-D to 1-D is shown in table 3.1 below.

Table 1. Mapping Table from 3-D to 1-D.

Point	Dimensions			Bit	Decimal code
	0	1	2		
(0,1,2)	000	001	010	000001010	10
(2,1,4)	010	001	100	001100010	98
(0,0,7)	000	000	111	001001001	73
(7,0,7)	111	000	111	101101101	365
(7,4,2)	111	100	010	110101100	428

The evaluation results and comparison with different algorithms for CPU scheduling in [5, 12] show that the CPU utilization of our algorithm (SFCP) is maximum and success ratio is better.

V. DISK-SCHEDULING ALGORITHMS BASED ON SPACE-FILLING CURVES

The problem of scheduling a set of tasks with time and resource constraints is known to be NP-complete [13]. Several heuristics have been developed to approximately optimize the scheduling problem. Traditional disk scheduling algorithms [14] are optimized for aggregate throughput. These algorithms, including SCAN, LOOK, C-SCAN, and SATF (Shortest Access Time First), aim to minimize seek time and/or rotational latency overheads.
They offer no QoS assurance other than perhaps absence of starvation. Deadline-based scheduling algorithms [13, 15, 16] have built on the basic earliest deadline first (EDF) schedule of requests to ensure that deadlines are met. These algorithms, including SCAN-EDF and feasible-deadline EDF, perform restricted reorderings within the basic EDF schedule to reduce disk head movements while preserving the deadline constraints. Like previous work on QoS-aware disk scheduling, space-filling curves explicitly recognize the existence of multiple and sometimes-antagonistic service objectives in the scheduling problem.
A more general model of mapping service requests in the multi-dimensional space into a linear order that balances between the different dimensions is given [4, 5]. Disk schedulers based on space-filling curves generalize traditional disk schedulers.
In the QoS-aware disk scheduler, a disk request is modeled by multiple parameters, (e.g., the disk cylinder, the real-time deadline, the priority, etc.) and represented as a point in the multi-dimensional space where each parameter corresponds to one dimension. Using a space-filling curve, the multi-dimensional disk request is converted to a one-dimensional value. Then, disk requests are inserted

into a priority queue according to their one-dimensional value with a lower value indicating a higher priority. Figure 5 gives an illustration of an SFC-based disk scheduler**.**

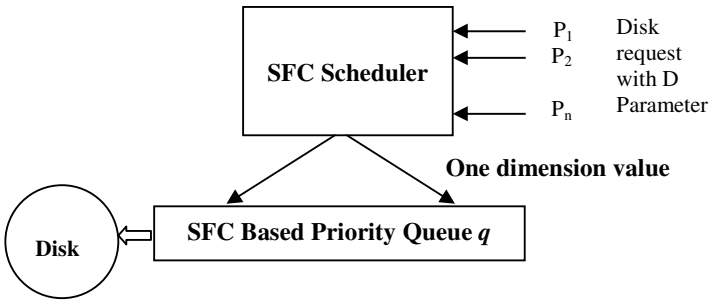

Figure 5. SFC-based disk scheduler

A new conditionally-preemptive disk scheduling algorithm is proposed [5] with SFC which trade-off between the fully-preemptive and the non-preemptive disk schedulers. In the conditionally-preemptive disk-scheduling algorithm, a newly arrived disk request T_{new} preempts the process of walking through a full cycle if and only if it has significantly higher priority than the currently served disk request.
In [3] describes many benefits of SFC in disk scheduling minimizing priority inversion, avoiding starvation, effective disk utilization in context of real-time constraints based request by considering other parameter associated with request.

VI. CONCLUSION

In this paper, we describe and review space-filling curves. Space-filling curve techniques have certain unique properties like map the multiple QoS parameters into the one-dimensional space. These properties have been used in recently for scheduling CPU transaction and disk request in real-time environment. Also their mapping and advantages are explored. Our brief description and study about SFC, we say that SFC can further be used in many more application area just like scheduling task in real-time operating system where each task is having its own important associated with multiple parameter or dimension.

REFERENCES

[1] Hans Sagan, “Space-Filling Curves”, New York, Springer-Verlag, 1994. ISBN: 0-387-94265-3.

[2] Mohamed F. Mokbel, Walid G. Aref and Ibrahim Kamel, “Performance of Multi-Dimensional Space-filling Curves”, in Proceedings of the 10th ACM international symposium on Advances in Geographic Information Systems, McLean, Virginia, USA, pp. 149-154, 2002.

[3] Mohamed F. Mokbel, Walid G. Aref, Khaled Elbassioni and Ibrahim Kamel, “Scalable Multimedia Disk Scheduling”, in Proceedings of the 20th International Conference on Data Engineering, pp. 498-509, 30 March-02 April 2004.

[4] M. Ahmed and S. Bokhari, “Mapping with Space Filling Surfaces”, IEEE Transactions on Parallel and Distributed Systems, volume 18, issue 09, pp. 1258-1269, September 2007.

[5] M. F. Mokbel and W. G. Aref. “Irregularity in Multi-Dimensional Space-Filling Curves with Applications in Multimedia Databases”, in the Proceedings of the 10th International Conference on Information and Knowledge Management, CIKM, Atlanta, Georgia, USA, pp. 512-519, November 2001.

[6] C. W. Ou, M. Gunwani, and S. Ranka, “Architecture-Independent Locality-Improving Transformations of Computational Graphs Embedded in k-Dimensions,” in the Proceeding Ninth ACM International Conference on Super-computing, pp. 289-297, July 1995.

[7] S. Aluru and F. Sevilgen, "Parallel Domain Decomposition and Load Balancing Using Space-Filling Curves," in the Proceeding Fourth IEEE International Confernce on High Performance Computing, pp. 230-235, 1997.

[8] S. Chatterjee, A. Lebeck, P. Patnala and M. Thottethodi, "Recursive Array Layouts and Fast Parallel Matrix Multiplication," in the Proceeding of Annual ACM Symposium Parallel Algorithms and Architectures (SPAA), pp. 222-231, 1999.

[9] Y. Zhu and Y. Hu, "Efficient, Proximity-Aware Load Balancing for DHT-Based P2P Systems," IEEE Trans. Parallel and Distributed Systems, vol. 16, no. 4, pp. 349-361, Apr. 2005.

[10] H. V. Jagadish, "Analysis of the Hilbert Curve for Representing Two-Dimensional Space," Information Processing Letters, vol. 62, pp. 17-22, 1997.

[11] B. Moon, H. V. Jagadish, C. Faloutsos, and J. H. Saltz, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," IEEE Transaction on Knowledge and Data Engineering, vol. 13, no. 1, pp. 124-141, Jan.-Feb. 2001.

[12] G. R. Bamnote & Dr. M. S. Ali, "Resource Scheduling in Real-time Database Systems" PhD Thesis, Sant Gadge Baba Amravati University, Amravati, 2009.

[13] Ben Kao and Hector Garcia-Molina, "An Overview of Real-Time Database Systems", in Proceedings of NATO Advanced Study Institute on Real-Time Computing, St. Maarten, Netherlands Antilles, Springer-Verlag, October 1992.

[14] A. Silberchatz and P. Galvin. Operating System Concepts. Addison-Wesley, 5th edition, 1998.

[15] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions: A Performance Evaluation", in Proceedings of the 14th International Conference on Very Large Data Bases, Los Angeles, California, pp. 01-12, March 1988.

[16] R. Abbott and H. Garcia-Molina, "Scheduling Real-Time Transactions with Disk Resident Data", in Proceedings of 15th International Conference on Very Large Databases, pp. 385-396, August 1989.

Authors

Mir Ashfaque Ali is Head of Information Technology Department at Government Polytechnic Amravati, Maharashtra, India. He did M.S in Computer Science and B.E in Computer Engineering. He has 20 years of teaching experience.

S. A. Ladhake is Principal in Sipna's College of Engineering & Technology, Amravati, Maharashtra, India. He is PhD, ME (Electronics), P.G.D.I.T. He is having 28 Yrs. teaching Experience. He is member of professional bodies FIETE, MIEEE, FIE, MISTE.