Space Filling Curve Partitioning - A Literature Survey

I. ABSTRACT

Space-filling curves (SFCs) are commonly used as a linear mapping of a multi-dimensional space. As data held in main memory or an external storage must be of a single dimension, every application involving multi-dimensional data must be mapped to a one-dimensional domain. Space-filling curves are effective at indexing these multi-dimensional applications due to their faculty for preserving the locality of objects from higher dimensions in linear space.

This paper presents a survey on the effectiveness of partitions produced by space-filling curves for parallelisation algorithms, whilst providing a brief history behind space-filling curves and looking at their relevance in the modern world.

II. DEFINITIONS

Data acquisition: The need to communicate between processors so that they have the data they need to carry out the necessary computations on their subset of the data [15].

Decomposition: The strategy used to partition a problem amongst the available processors [15].

Hölder continuous: A mapping $f: I \to \mathbb{R}^n$ is called Hölder continuous with exponent r on the interval I, if there is a constant $C > 0$ for all parameters $x, y \epsilon I$, such that [2]:

$$\|f(x) - f(y)\|_2 \leq C|x - y|^r$$

Hypercube: A geometric figure in Euclidean space of n dimensions that is analogous to a cube in three dimensions [9].

Locality: Points close in $[N]^m$ are also close in the traversal order, and vice versa [4].

Parallelisation: The process of partitioning a computational domain into equal parts and distribute these partitions onto the available computing cores [2].

Partition: A partition of a set X is a set of nonempty subsets of X such that every element x in X is in exactly one of these subsets [7].

Quadtree: A tree structure in which each non-leaf node has four children [16].

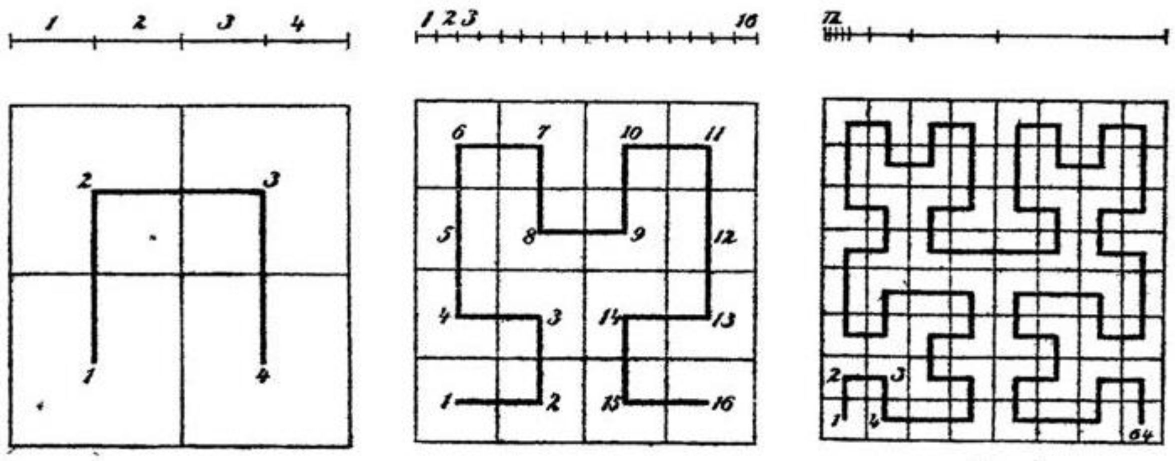

Figure 1. The first three iterations of the Hilbert curve. [8]

Space-filling curve: A continuous one-to-one mapping of $[0,1]$ into $[0,1]^n$ [14].

Surjective map: Let f be a function defined on a set A and taking values in a set B. Then f is said to be a surjective map if, for any b in B, there exists an a in A for which $b = f(a)$ [18].

III. INTRODUCTION

Is it possible to obtain a continuous surjective mapping from $[0,1]$ onto $[0,1]^2$?

In 1890, Giuseppe Peano [12] solved this question by producing such a curve in mathematical terms. The following year, David Hilbert [8] found a variant of the solution and constructed a geometric representation of the curve, shown in Figure 1. Today, these curves are categorised as space-filling curves and conform with the following definition:

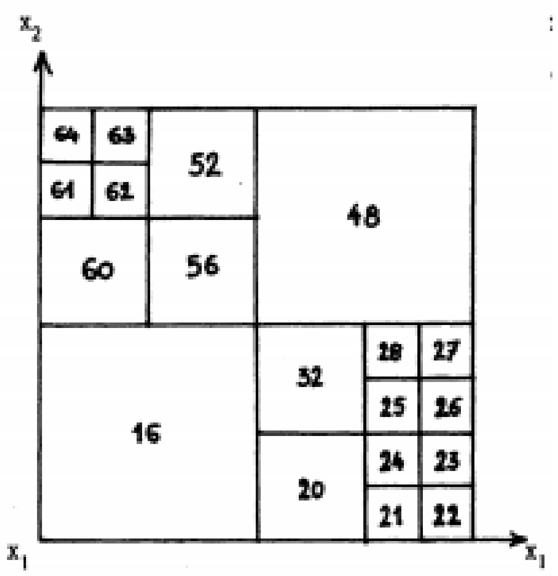

Figure 2. A scanning algorithm based on the Hilbert curve indexing a quadtree grid. [14]

Given a curve $f_*(I)$, where $I \subset \mathbb{R}$ and the corresponding mapping $f: I \to \mathbb{R}^n$, then $f_*(I)$ is called a space-filling curve, if $f_*(I)$ has a strictly positive Jordan content (area, volume, …)[2].

Implicitly, space-filling curves can therefore index hypercubes. Taking the base, two-dimensional case, Quinqueton and Berthod [14] took the Hilbert curve one step further, designing an adaptive space-filling scanning algorithm able to index Cartesian grids following a quadtree structure. An example execution of the algorithm is shown in Figure 2 and Bader shows how this can be applied to geometric models in Figure 3. Evidently, it is possible to cut the curve and divide the grid into segments. Therefore, the surface to volume ratio of the segments created by a partition can be calculated on adaptive grids.

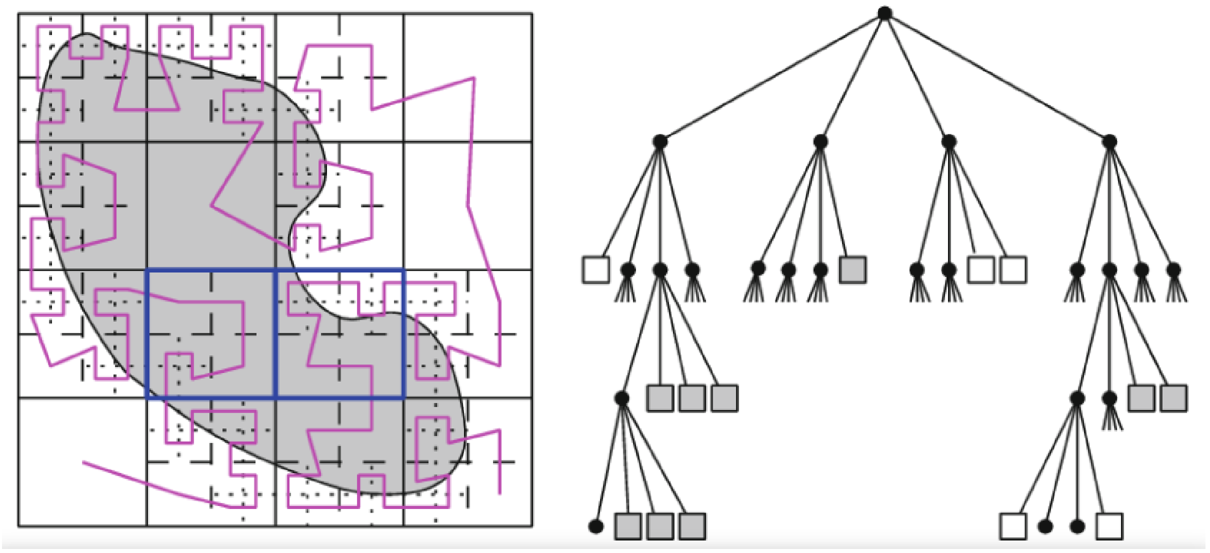
Figure 3. Quadtree representation of a given object and a sequential order on the quadtree cells that avoids jumps – the order corresponds to a depth-first traversal of the corresponding tree [2].

IV. QUASI-OPTIMAL PARTITIONS

In 1997, Parashar et al. [11] introduced a technique using space-filling curves to parallelise general partial differential equation (PDE) solvers. Researching this further, Griebel and Zumbusch [6] implemented a multigrid method for parallelising PDE solvers and analysed the quality of partitions produced by space-filling curves [5].

In the paper, the execution time of a parallel program was considered dependent on the parallel computing time and the communication time between the processors.

Considering $\mathrm{O(n)}$ algorithms, the parallel computing time is calculated from the partition of n data onto p processors. Given that the data to be transferred across the processors is proportional to the surface s of the partition, the following formula for runtime was formulated:

$$t = C_1\frac{n}{p} + C_2(t_{startup} + s * t_{bandwidth})$$

Accordingly, the key to the efficiency of the partition was to minimise the surface to volume ratio $\frac{s}{v}$, where $v = \frac{n}{p}$, since rearranging the formula gives:

$$efficiency = \frac{1}{1+\frac{C_2}{C_1}(\frac{1}{v}t_{startup} + \frac{s}{v} * t_{bandwidth})}$$

Explaining that the n-sphere holds the lowest continuous surface to volume ratio, Griebel and Zumbusch generalised the equation and regarded estimates of type $s \leq C_{part} \cdot v^{\frac{d-1}{d}}$ with low constants C_{part} as optimal. With Gotsman and Lindenbaum [4] proving that the locality of a d-dimensional Hilbert curve was Hölder continuous and held an upper bound of $(d+3)^{\frac{d}{2}}2d$, Griebel and Zumbusch derived a lemma followed by a corollary explaining how the surface of the partition generated from a space-filling curve is also bounded by $C_{part} \cdot v^{\frac{d-1}{d}}$ and furthermore, can be applied to the partitioning of quasi-uniform meshes. As a result, the efficiency of partitions based on space-filling curves is as follows:

$$efficiency = \frac{1}{1+\frac{C_2 C_{part} t_{bandwith}}{C_1} \cdot \frac{p}{n^{\frac{1}{d}}}}$$

Therefore, conclusively, Griebel and Zumbusch demonstrated that space-filling curves achieve optimal parallel efficiency as $n \to \infty$ on regular grids. Whether a quantitative formula exist for adaptive grids is yet to be solved. If found however, Griebel and Zumbusch's findings can be used to measure the efficiency of the partitions created.

V. APPLICATIONS

In 1986, quadtrees and octrees acquired attention following the presentation of the Barnes-Hut algorithm [3], used to perform an approximation of an n-body simulation. Looking for a parallelised solution to the n-body simulation, Warren et al. [15, 17] turned to space-filling curves for domain decomposition. Using the Z-order curve for partitioning, Warren et al. noted that the spatial locality in the decomposition reduced the amount of inter-processor communication traffic.

In 1997, Aluru and Sevilgen [1] highlighted the three key factors behind effective partitions for parallelisation: balancing workload, minimising inter-processor communication and minimising the running time of the partitioning algorithm. Assisted by Ou et al. [10], Pilkington and Baden [13], who showed that space-filling create highly efficient partitions at very low costs, Aluru and Sevilgen demonstrated a method for ordering and partitioning dynamic, multi-dimensional data across p processors using space-filling curves.

Today, space-filling curves are a favourable method of partitioning multi-dimensional data for parallelisation and can be found in a multitude of various applications [2].

VI. CONCLUSION AND OUTLOOK

Space-filling curves present an effective method of mapping multidimensional data to one dimension for parallel computing. When dividing such curves into equally sized fragments on regular underlying grids, the process is known to yield quasi-optimal partitions. Griebel and Zumbusch [5] demonstrated this efficiency by proving that the partitions generated were as efficient as n-spheres besides a constant when measuring their surface relative to the volume. Simplistically, the lower the ratio, the less inter-processor communication necessary. However, it is unknown whether this applies to adaptive grids and further research would be necessary in finding whether a quantitative formula exists for adaptive grids.

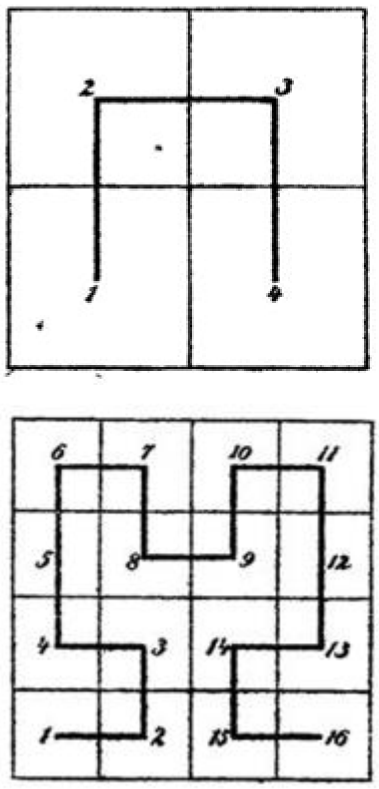

Figure 4. The first two iterations of the Hilbert curve taken from figure 1.

If a brute-force approach were to investigate this further, measures would need to be taken to lower the time complexity. To partition the initial two-dimensional Hilbert curve shown in figure 4 into three segments, one segment would be required to cover two cells and the other two segments would cover one cell e.g. [1,1,2,3]. Shifting the two through the array, four unique partitions are created. Comparatively, looking at the second iteration of the Hilbert curve, there are sixteen cells, giving $\frac{3\cdot(\frac{16-1}{3})^2-(\frac{16-1}{3})}{2}\cdot 16 = 35\cdot 16 = 450$ unique possible partitions. This explosion in partition opportunities within the first two iterations of the Hilbert curve suggests methods of culling the sample size for each grid generated would be necessary in retrieving the surface-to-volume ratio of adaptive grids featuring more than sixteen cells.

References:

[1] S. Aluru and F. E. Sevilgen, "Parallel Domain Decomposition and load balancing using space-filling curves. In *Proceedings of the Fourth International Conference on High Performing Computing*, pages 230-235. IEEE Computer Society, 1997.

[2] M. Bader, "Space-filling curves: an introduction with applications in scientific computing." *Springer*, 9:17-163, 2013.

[3] J. Barnes and P. Hut, “A hierarchical O(N log N) force-calculation algorithm.” *Nature*, 324: 446–449, 1986.

[4] C. Gotsman and M. Lindenbaum, “On the metric properties of discrete space-filing curves.” *IEEE Transactions on Image Processing*, 5(5):794–797, 1996.

[5] M. Griebel and G. Zumbusch, “Hash based adaptive parallel multilevel methods with space-filling curves.” In H. Rollnik and D. Wolf, editors, NIC *Symposium* 2001, volume 9 of NIC *Series*, pages 479-492, Forschungszentrum Jülich, 2002.

[6] M. Griebel and G. Zumbusch, “Parallel multigrid in an adaptive PDE Solver based on hashing and space-filling curves.” *Parallel Computing*, 27(7):827-843, 1999.

[7] P. Halmos, “Naive Set Theory.” *Springer*, p. 28, 1960

[8] D. Hilbert, “Ueber die stetige Abbildung einer Linie auf ein Flächenstück.” *Mathematische Annalen*, 38:459-460, 1891.

[9] "Hypercube." From Dictionary by Merriam-Webster. https://www.merriam-webster.com/dictionary/hypercube

[10] C. W. Ou, S. Ranka and G. Fox, “Fast and parallel mapping algorithms for irregular problems.” Journal of Supercomputing, 10:119-140, 1996.

[11] M. Parashar, J. C. Browne, C. Edwards and K. Klimkowski, “A Common Data Management Infrastructure for Adaptive Algorithms for PDE Solutions.” In *Proceedings of the 1997 ACM/IEEE Conference on Supercomputing*, pages 1-22. ACM Press, 1997.

[12] G. Peano, “Sur une courbe, qui remplit toute une aire plane.” *Mathematische Annalen*, 36(1): 157–160, 1890.

[13] J. R. Pilkington and S.B. Baden, “Dynamic partitioning of non-uniform structured workloads with space-filling curves.” *IEEE Transaction on Parallel and Distributed Systems*, 7(3):288-300, 1996.

[14] J. Quinqueton and M. Berthod, “A locally adaptive Peano scanning algorithm.” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-3(4):403-412, 1981.

[15] J. K. Salmon, M. S. Warren and G. S. Winckelmans, “Fast parallel tree codes for gravitational and fluid dynamical N-body problems.” *International Journal of High Performance Computing Applications*, 8(2), 1994.

[16] H. Samet, “The quadtree and related hierarchical data structures.” *Computing Surveys*, 16(2):187–260, 1984.

[17] M. S. Warren and J. K. Salmon, “A parallel hashed oct-tree n-body algorithm.” In *Conference on High Performance Networking and Computing, Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*, pages 12-21. ACM, 1993.

[18] Eric W. Weisstein, "Surjection." From MathWorld - A Wolfram Web Resource. https//mathworld.wolfram.com/Surjection.html