Load Balancing using Hilbert Space-filling Curves

Hui Liu

1 Introduction

Parallel computers, which are a group of computers (nodes) connected by high speed networks, such as InfiniBand, have been employed to accelerate parallel applications. MPI (Message Passing Interface) is commonly accepted as a communication environment. It is well-known that the scalability of a parallel application is determined by communication and the nature of the application, such as the proportion of execution time that can be parallelized in the application. The keys are to identify the parallelizable portion and to reduce communication among these nodes. For grid-based methods, such as the finite element, finite volume and finite difference methods, the communication pattern of a parallel application can be determined by a given grid and its numerical method, and it can be described by a graph. Many graph-based grid partitioning algorithms have been proposed, including spectral methods [5, 7], multilevel methods [8, 9], diffusive methods [10, 11], and refinement-tree methods, whose partitioning quality is excellent. These algorithms have been implemented and are available online. METIS and ParMETIS [2, 12], Scotch and PTScotch [13, 14], Chaco [15] and Jostle [16] are famous graph partitioning packages. However, one disadvantage of the graph methods is that they are complex, and they are hard to implement if a load balancing module must be implemented. As an alternative, geometric methods have been studied, which assume that objects have a higher chance to communicate with nearby objects [1]. Their partitioning quality is also good and easier to implement, and they are good alternatives for graph methods. Therefore, geometric methods are also popular in parallel computing. Many geometric algorithms have been developed for them, including space-filling curves (Hilbert space-filling curves and Morton space-filling curves [1]), recursive bisection [4, 5, 6], and Zoltan [3].

2 Space-filling Curves and Orders

Space-filling curves are those curves that fill an entire n-dimensional unit hypercube, which were proposed by Peano in 1890 and popularized by Hilbert later [17, 27].

Many space-filling curves have been discovered. Figures 1 and 2 show levels 1, 2, 3 and 6 Hilbert space-filling curves in a two-dimensional unit square. It can be seen that a curve is denser if a level is higher. Figure 3 shows levels 1, 2 and 3 Sierpiński space-filling curves. Figure 4 shows a level 2 Morton space-filling curve. These curves show that the Hilbert space-filling curves and the Sierpiński space-filling curves have good locality (nearby points in the multiple dimensional space are mapped to nearby points in one-dimensional space [17, 28]), and the Morton space-filling curves have jumps, whose locality is poor.

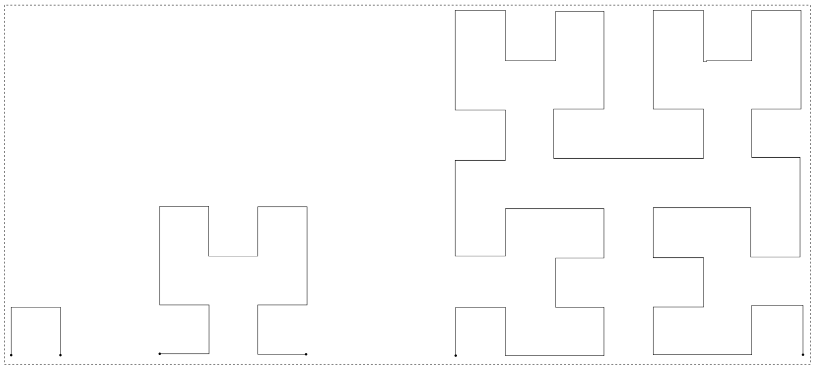

Figure 1: Hilbert space-filling curves, two dimensions, levels 1, 2 and 3

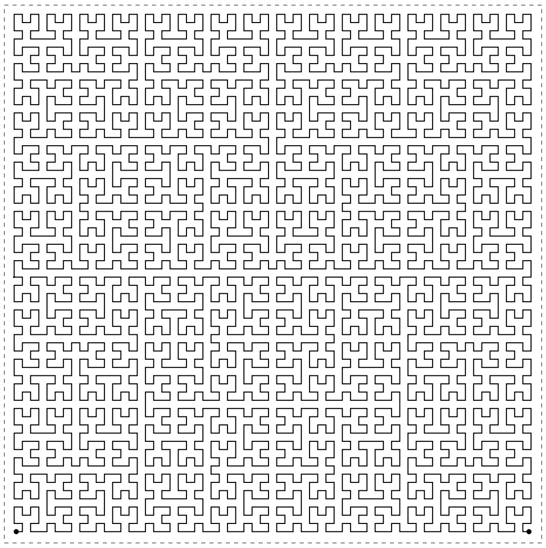

Figure 2: Hilbert space-filling curve, level 6

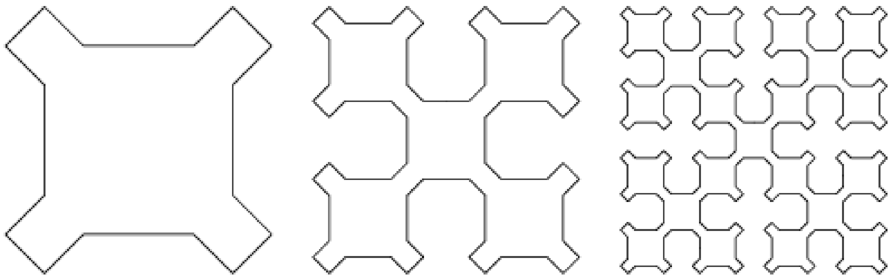

Figure 3: Sierpiński space-filling curves, levels 1, 2 and 3

2.1 Hilbert Space-filling Curve Orders

Each curve has a starting point and an ending point. Along this curve, a map is introduced between a one-dimensional domain and a multi-dimensional domain. Figures 5 and 6 show a level 2 Hilbert space-filling

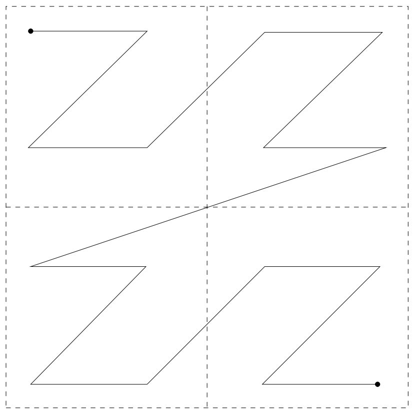

Figure 4: Morton space-filling curve, level 2

curve and a level 2 Morton space-fill curve, respectively. Both of them have 16 vertices whose indexes start from 0 to 15, which means that these curves define orders and they map a two-dimensional space to a one-dimensional space. Higher dimensional space-filling curves are defined similarly.

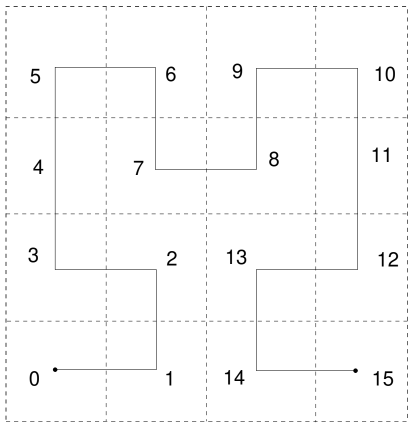

Figure 5: Hilbert space-filling curve order, level 2

A Hilbert space-filling curve [17] has many important characteristics, such as locality and self-similarity (an object is self-similar if small portions of it are reproductions of initial objects. A Hilbert curve (order) has been applied in many areas, including image storing, database indexing, data compression and load balancing. For parallel computing, the Hilbert order method is one of the most important geometry-based partitioning methods.

Algorithms for computing Hilbert curves and Hilbert space-filling curve orders in two- and three-dimensional spaces have been proposed in the literature, which can be classified into recursive algorithms [18, 19, 20, 21] and iterative algorithms [22, 1, 23, 24, 25, 26]. Iterative algorithms, especially the table-driven algorithms [22, 1], are usually much faster than recursive algorithms. In general, the complexities of these algorithms are $O(m)$, where m is the level of a Hilbert curve. For a two-dimensional space, Chen et al. [26] proposed an algorithm of $O(r)$ complexity, where r is defined as $r = \log_2(\max(x,y)) + 1$, $r \leq m$ and is independent of the level m. This algorithm is faster when m is much larger than r. The same idea

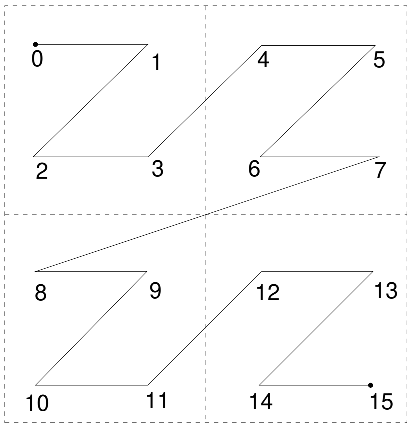

Figure 6: Morton space-filling curve order (Z-order), level 2

was also applied to a three-dimensional space.

3 Space-filling Curve Partitioning Method

A space-filling curve defines a map from a multi-dimensional space to a one-dimensional space. Since Hilbert space-filling curves have excellent locality, they are good alternatives for load balancing methods. Algorithm 1 shows the process of the space-filling curve methods, which has three steps:

1. The first step is to map the given computational domain Ω to a subset of $(0,1)^3$ (boundaries are excluded), which can be obtained by a linear mapping. In this case, for any cell, the coordinate used to represent the cell belongs to $(0,1)^3$. In Algorithm 1, when the computational domain is mapped, its aspect ratio is preserved.
2. A space-filling curve is employed to convert coordinates from $(0,1)^3$ to a value in set $(0,1)$.
3. The third step is to partition $(0,1)$ into N_p sub-intervals such that each sub-interval has the same number of cells (or workload).

Each MPI process can perform the first and the second steps independently. In our implementation, the third step is calculated by one MPI process and it broadcasts results to other MPI processes. Also, different space-filling curves define different partitioning methods. The space-filling curve methods assume that two cells (elements) that are close to each other have a higher possibility to communicate with each other than two cells that are far from each other. This assumption is true for grid-based numerical methods, such as the finite element, finite volume and finite difference methods. The only difference for various space-filling curve methods is how to map a cell to $(0,1)$.

References

[1] P. M. Campbell and K. D. Devine and J. E. Flaherty and L. G. Gervasio and J. D. Teresco, Dynamic load balancing using space-filling curves, Technical Report CS-03-01, 2003.

Algorithm 1 Space-filling curve method

```
1: Map a computational domain $\Omega$ to a subset of $(0,1)^3$.
2: For any cell, calculate its mapping that belongs to $(0,1)$.
3: Partition the interval $(0,1)$ into $N_p$ sub-intervals such that each sub-interval has the same number of
   cells.
```

[2] George Karypis and Kirk Schloegel and Vipin Kumar, PARMETIS: Parallel Graph Partitioning and Sparse Matrix Ordering Library version 3.1, 2003.

[3] E. Boman and K. Devine and R. Heaphy and B. Hendrickson and V. Leung and L.A. Riesen and C. Vaughan and U. Catalyurek and D. Bozdag and W. Mitchell and J. Teresco, Zoltan v3: Parallel Partitioning, Load Balancing and Data-Management Services, User's Guide, Sandia National Laboratories Tech. Rep. SAND2007-4748W, 2007.

[4] M. J. Berger, S. H. Bokhari, A partitioning strategy for nonuniform problems on multiprocessors, IEEE Trans. Computers, 36(5) (1987) 570-580.

[5] H. D. Simon, Partitioning of unstructured problems for parallel processing, in: Proc. Conference on Parallel Methods on Large Scale Structural Analysis and Physics Applications, Pergammon Press, 1991.

[6] V. E. Taylor, B. Nour-Omid, A study of the factorization fill-in for a parallel implementation of the finite element method, Int. J. Numer. Meth. Engng. 37 (1994) 3809-3823.

[7] A. Pothen, H. Simon, K. Liou, Partitioning sparse matrices with eigenvectors of graphs, SIAM J. Matrix Anal. 11 (3) (1990) 430-452.

[8] T. Bui, C. Jones, A heuristic for reducing fill in sparse matrix factorization, in: Proc. 6th SIAM Conf. Parallel Processing for Scientific Computing, SIAM, 1993, pp. 445-452.

[9] G. Karypis, V. Kumar, A fast and high quality multilevel scheme for partitioning irregular graphs, Tech. Rep. CORR 95-035, University of Minnesota, Dept. Computer Science, Minneapolis, MN (June 1995).

[10] G. Cybenko, Dynamic load balancing for distributed memory multiprocessors, J. Parallel Distrib. Comput. 7 (1989) 279-301.

[11] E. Leiss, H. Reddy, Distributed load balancing: design and performance analysis, W.M. Keck Research Computation Laboratory 5 (1989) 205-270.

[12] George Karypis and Vipin Kumar, A Parallel Algorithm for Multilevel Graph Partitioning and Sparse Matrix Ordering, Journal of Parallel and Distributed Computing, Vol. 48, pp. 71 - 85, 1998

[13] F. Pellegrini, static mapping by dual recursive bipartitioning of process and architecture graphs, Proceedings of SHPCC'94, Knoxville, Tennessee, pages 486-493, IEEE Press, May 1994.

[14] C. Chevalier, F. Pellegrini, PT-Scotch: A tool for efficient parallel graph ordering, Parallel Computing, Volume 34, Issues 6-8, July 2008, Pages 318-331.

[15] Bruce Hendrickson, Chaco, Encyclopedia of Parallel Computing, 2011, 248-249, Springer US.

[16] Christopher Walshaw and Mark Cross, JOSTLE: multilevel graph partitioning software: an overview, Mesh partitioning techniques and domain decomposition techniques, Saxe-Coburg Publications, Stirling, Scotland, UK, pp. 27-58, 2007.

[17] H. Sagan, Space-Filling Curves. Springer-Verlag; 1994.

[18] A. R. Butz, Altrnative algorithm for Hilbert's space-filling curve. IEEE Transactions on Computers 1971; 20: 424–426.

[19] L. M. Goldschlager, Short algorithms for space-filling curves. Software—Practice and Experience 1981; 11: 99–100.

[20] I. H. Witten, B. Wyvill, On the generation and use of space-filling curves. Software—Practice and Experience 1983; 13: 519–525.

[21] A. J. Cole, A note on space filling curves. Software—Practice and Experience 1983; 13: 1181–1189.

[22] J. G. Griffiths, Table-driven algorithms for generating space-filling curves. Computer-Aided Design 1985; 17(1): 37–41.

[23] X. Liu, G. F. Schrack, Encoding and decoding the Hilbert order. Software—Practice and Experience 1996; 26(12): 1335–1346.

[24] X. Liu, G. F. Schrack, An algorithm for encoding and decoding the 3-D Hilbert order. IEEE transactions on image processing 1997; 6: 1333–1337.

[25] A. J. Fisher, A new algorithm for generation hilbert curves. Software: Practice and Experience 1986; 16: 5–12.

[26] N. Chen, N. Wang, B. Shi, A new algorithm for encoding and decoding the Hilbert order. Software—Practice and Experience 2007; 37(8): 897–908.

[27] C. Li, Y. Feng, Algorithm for analyzing n-dimensional Hilbert curve , vol. 3739. Springer Berlin/Heidelberg, 2005; 657–662.

[28] Nico Reissman, Jan Christian Meyer, Magnus Jahre, A Study of Energy and Locality Effects Using Space-Filling Curves, Proceedings of the 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, 815-822, May 19 - 23, 2014.