A Formal Analysis of Space Filling Curves for Parallel Domain Decomposition

Srikanta Tirthapura, Sudip Seal and Srinivas Aluru
Department of Electrical and Computer Engineering
Iowa State University, Ames, IA
{snt, skseal, aluru}@iastate.edu

Abstract

Space filling curves (SFCs) are widely used for parallel domain decomposition in scientific computing applications. The proximity preserving properties of SFCs are expected to keep most accesses local in applications that require efficient access to spatial neighborhoods. While experimental results are used to confirm this behavior, a rigorous mathematical analysis of SFCs turns out to be rather hard and rarely attempted. In this paper, we analyze SFC based parallel domain decomposition for a uniform random spatial distribution in three dimensions. Let n denote the expected number of points and P denote the number of processors. We show that the expected distance along an SFC to a nearest neighbor is $O(n^{2/3})$. We then consider the problem of answering nearest neighbor and spherical region queries for each point. For $P = n^{\alpha}$ ($0 < \alpha \leq 1$) processors, we show that the total number of remote accesses grows as $O(n^{3/4+\alpha/4})$. This analysis shows that the expected number of total remote accesses is sublinear for any sublinear number of processors. We view the analysis presented here as a step towards the goal of understanding the utility of SFCs in scientific applications and the analysis of more complex spatial distributions.

Key words: domain decomposition, parallel algorithms, probabilistic analysis, space filling curves.

1. Introduction

Many applications in scientific computing deal with entities (particles, grid cells, unknowns) in three dimensional space whose interactions are limited to spatial neighborhoods. In grid based methods such as multigrid and adaptive mesh refinement, information flow occurs between adjoining grid cells, or from parent cells to daughter cells and vice versa. In graphs resulting from the finite element method, graph edges exist only between nodes whose corresponding elements share physical boundaries. In particle based methods such as molecular dynamics and smoothed particle hydrodynamics, particle interactions are restricted to spatial neighborhoods. The same is true for local field computation in the fast multipole method and its many applications. A common theme underlying all these methods is the execution of a large number of time steps during each of which interactions are computed as prescribed by the particular method.

Without loss of generality, we refer to point data throughout this paper. Points represent locations of particles in particle based methods, locations of atoms in molecular dynamics, and spatial locations of unknowns in some other methods. The centers of grid cells can serve the same purpose in grid based methods. Similarly, a convenient reference to points can be chosen for finite element decompositions, for example, locations of centroids for triangular elements. Parallelizing the aforementioned applications requires distributing spatial data to processors in a load balanced manner such that information required for computing interactions is most often locally satisfied within each processor. This can be formalized as balanced partitioning of the interaction graph while minimizing cross edges. Graph partitioning is known to be NP-hard and several heuristic methods and graph partitioning tools have been designed.

Any partitioning of spatial point data can be viewed as imposing a one-dimensional order based on processor indices and any ranking of points within the same processor. The goal is to have as much of the spatial neighborhood of a point as possible on the processor to which it is assigned. Therefore, one can ask the question: which one dimensional ordering of the underlying spatial data best preserves the spatial proximity relevant to the application at hand? This is why space filling curves (SFCs) are seen as effective in domain decomposition for scientific applications. A significant advantage of using SFC orderings is that they can be computed very fast, especially when compared to graph partitioning techniques. Besides, a simple scheme determines which processor should have a point based on its spatial location, without explicitly needing to store edges of the interaction graph.

Space filling curves are used in diverse applications

ranging from databases [7] to processor allocation strategies on multiprocessor systems [5]. They are frequently used in many scientific applications including multigrid methods [3], adaptive mesh refinement [11], fast multiple method [12] and molecular dynamics [9].

While a few formal results on locality preserving properties of SFCs are available (for example, see [2, 7]), the use of SFCs in scientific applications is often empirically justified by presenting good scaling and run-time results (for example, see [1, 10, 11]). Though such evidence is extremely useful and practical, a formal analysis will provide a better fundamental understanding and can potentially lead to valuable insights. This is the motivation behind the work presented in this paper. We limit our analysis to the case of uniform random spatial distributions and address the following two queries:

- **Nearest neighbor queries:** Given a set of n points, find all nearest neighbors of each point.
- **Spherical region queries:** Given a set of n points and a cutoff radius r, find all the points that lie within distance r from each point.

These queries are used in many scientific applications. For example, spherical region queries are used in molecular dynamics simulations for estimating the rapidly decaying Lennard-Jones potential.

Consider a set of points uniformly distributed in a $2^k \times 2^k \times 2^k$ decomposition of space, where each cell is occupied with probability p. Let $n = 2^{3k}p$ denote the expected number of points. We prove the following:

1. The expected distance along the SFC to a nearest neighbor of a point is $O\left(n^{2/3}\right)$.

Let $P = n^\alpha$ $(0 < \alpha \leq 1)$ be the number of processors. We estimate the number of points requiring remote accesses when answering nearest neighbor or spherical region queries in parallel.

2. For nearest neighbor queries, the total number of points requiring remote accesses is $O\left(n^{3/4+\alpha/4}\right)$.
3. For spherical region queries with a fixed radius r, the total number of points requiring remote accesses is $O\left(n^{3/4+\alpha/4}\right)$.

These results provide some interesting insights into the behavior of SFCs. The first result seems to indicate that it is counterproductive to have fewer than $O\left(n^{2/3}\right)$ points per processor, thus restricting the number of processors to $O\left(n^{1/3}\right)$, as the expected distance for each point will fall outside the purview of the processor otherwise. However, this conclusion is incorrect. The latter results show that for any sublinear number of processors ($P = n^\alpha; \alpha < 1$), the total number of remote accesses is sublinear. This shows that the expectation derived in the first result is skewed by a few points having their spatial neighborhood located very far away. The number of remote accesses is $\Omega\left(n^{3/4}\right)$, and increases only as $\sqrt[4]{P}$ as the number of processors P is increased. For n^α processors, the ratio of computation to communication is $O\left(n^{1/4(1-\alpha)}\right)$, showing that communication costs can be contained for sufficiently large values of n.

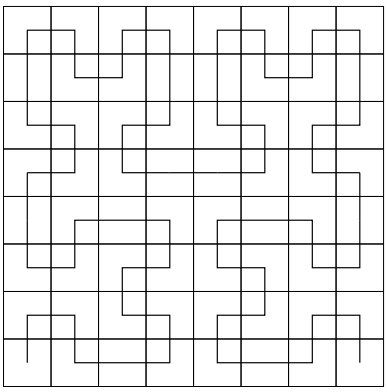

Figure 1. A 8×8 Hilbert space-filling curve [4] in two dimensions.

2. Modeling the Problem

Consider the decomposition of a three dimensional cube into $2^k \times 2^k \times 2^k$ cells. A space filling curve imposes a linear order on this array of cells. This can be depicted pictorially by connecting the cells in the SFC-order (see Fig. 1 for an example of Hilbert curve in 2-dimensions). SFCs are non-intersecting curves. Spatial point data is ordered using SFCs by choosing an appropriate decomposition and using SFC-order to order only the cells occupied by points.

Let $m = 2^{3k}$ denote the total number of cells. We populate the cells by independently considering each cell, and placing a point in it with probability p $(0 < p \leq 1)$. A cell is called *occupied* if a point is placed in it, and is called *unoccupied* otherwise. We assume the point in an occupied cell is placed at its center. Let σ denote the set of all 2^{3k} cells, and $\sigma(m, p)$ denote the set of occupied cells. A space filling curve is then used to arrange the cells in $\sigma(m, p)$ into a one dimensional array, called the *SFC-array*. Let $|\sigma(m, p)|$ denote the number of points. The problem size $n = E[|\sigma(m, p)|] = 2^{3k}p$.

Space filling curves are used for parallel domain decomposition by splitting the SFC-array evenly across processors using block decomposition. Let P denote the number

of processors. Each processor receives at most $\lceil \frac{|\sigma(m,p)|}{P} \rceil$ elements. We consider aggregate queries in which each point requires information about points in its spatial neighborhood, as defined by the particular query under consideration. If a query for a point can be answered using only the information available on the same processor, i.e., if the query point and the points that fall in the query region are all contained in the same processor, the query requires only *local access*. Otherwise, it is said to require *remote access*. We are interested in analyzing the number of queries requiring remote accesses.

In practice, SFC-array is computed as follows: The position of a cell in cell space can be described by integer coordinates (i, j, k) where $0 \leq i, j, k < 2^k$. The rank of a cell in SFC order can be computed by an SFC-specific mapping function that takes the integer coordinates as argument. The SFC-array is computed by sorting the ranks of occupied cells. The same can be achieved on a parallel computer using parallel sorting. Parallel sorting also has the desirable side effect of partitioning the SFC-array across processors. The SFC-array distance between two cells $u, v \in \sigma(m, p)$ is defined as the difference between the indexes of u and v in the SFC-array. The SFC-array distance between two points is defined as the SFC-array distance between the cells that contain the points.

3. Preliminary Results

In this section we derive some counting results on the number of cells contained in certain spherical and cubic regions, for use in subsequent analysis. The intuition behind why these results are useful is as follows: spheres are obviously relevant to spherical region queries. They naturally arise even in nearest neighbor queries because the sphere centered at a point with its nearest neighbor distance as radius will not have any points inside it. While spheres are relevant to capturing physical distances, analysis of a hierarchy of cubic regions is needed to relate the physical distances to SFC-array distances.

Definition 1 *Let $C(u, d)$ denote the region of overlap between a sphere of radius d centered at cell u and the $2^k \times 2^k \times 2^k$ array of cells.*

A cell v is said to be *contained* in $C(u, d)$ iff the center of v is contained within $C(u, d)$.

Lemma 1 *There are are least $\frac{d^3}{8}$ cells contained in $C(u, d)$, excluding u.*

Proof Consider a coordinate system with origin at the center of cell u. Irrespective of the location of u, we can find a point (x, y, z) such that $|x| = |y| = |z| = \frac{d}{\sqrt{3}}$ and (x, y, z) is on the surface of $C(u, d)$. Let $w = \lfloor \frac{d}{\sqrt{3}} \rfloor$. Since w is integral, the cube of side w with one corner at u and another corner at (x, y, z) has $(w + 1)^3$ grid cell centers inside, or on the boundary of the cube. But since $w + 1 > \frac{d}{\sqrt{3}} > d/2$, we have $(w + 1)^3 > d^3/8$, which proves the lemma. ∎

Note that a $2^k \times 2^k \times 2^k$ decomposition of a cube can be viewed as the result of partitioning the cube into 8 subcubes by bisecting along each dimension, and recursively carrying out the process on each of the subcubes $k - 1$ times. We use the term *standard cube* to refer to each intermediate cube resulting from this process. We will continue to use the term cell to describe cubes at the finest level of decomposition. When the cube is recursively decomposed $l \leq k$ times, there are 8^l standard cubes each containing 8^{k-l} cells. Each such cube is referred to as a cube at level l.

An important property of SFCs is that once they enter a standard cube, they leave it only after visiting all the cells in it. If the SFC order is viewed as an array, the collection of cells in a standard cube always corresponds to a subarray. Thus, the distance between two cells in the SFC-array is no more than the number of occupied cells in the smallest standard cube containing the cells.

Definition 2 *For cell u, let $A(u, d)$ denote the smallest standard cube that encloses the region $C(u, d)$.*

Let $|A(u, d)|$ denote the number of cells in $A(u, d)$. Clearly, $|A(u, d)|$ depends on the location of u. Its smallest value is $\Theta\left(d^3\right)$, corresponding to the cube of side length $2d$ enclosing $C(u, d)$. On the other hand, if u is adjacent to a boundary of the first recursive cut of the cube, then $|A(u, 1)| = 2^{3k}$. The following lemma shows that the average size of $A(u, d)$ is $O\left(dm^{\frac{2}{3}}\right)$.

Lemma 2 $\sum_{u \in \sigma} |A(u, d)| = O\left(d \cdot m^{\frac{5}{3}}\right)$

Proof Let S_l denote a cube at level l. For all the cells u located in a cuboid of dimensions $2d \times (2^{k-l} - 2d) \times (2^{k-l} - 2d)$ placed symmetrically at the center of S_l (see Fig. 2),

$A(u, d)$ is S_l. Hence, for each cell u within the region in S_l formed from the fusion of the three cuboids, $|A(u, d)| = 8^{k-l}$. The number of such cells in S_l is at most the total number of cells in the three cuboids, which is $3 \cdot 2d \cdot 2^{k-l} \cdot 2^{k-l} = 6 \cdot d \cdot 4^{k-l}$. If σ_l denotes the set of cells $u \in \sigma$ for which $A(u, d) = 8^{k-l}$, then $|\sigma_l| = 8^l \cdot (6 \cdot d \cdot 4^{k-l})$. Since for all $u \in \sigma_l$, $|A(u, d)| = 8^{k-l}$, it follows that:

$$\sum_{u \in \sigma} |A(u, d)| = \sum_{l=0}^{k} \sum_{u \in \sigma_l} |A(u, d)|$$

$$= \sum_{l=0}^{k} (8^l) \cdot (6 \cdot d \cdot 4^{k-l}) \cdot (8^{k-l}) = O\left(d \cdot m^{\frac{5}{3}}\right)$$ ∎

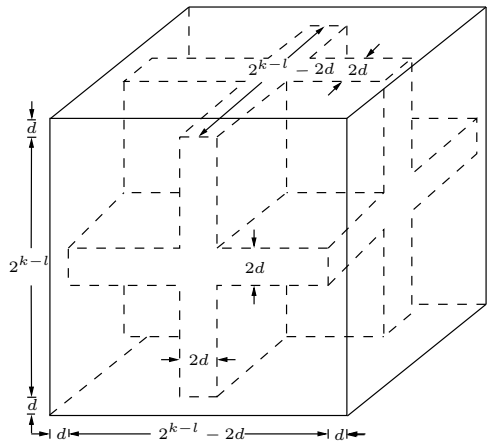

Figure 2. The bounding cube is S_l. The dashed inner region is composed from fusing three cuboids, each of dimensions $2d \times (2^{k-l} - 2d) \times (2^{k-l} - 2d)$. For a cell u in this region, $A(u, d)$ is S_l.

4. Nearest Neighbor Distance

In this section, we bound the average SFC-array distance between a point and its nearest neighbor. This is useful in formally characterizing the extent to which SFCs preserve locality. Some of the lemmas presented in this section are also subsequently used in analyzing the effectiveness of SFC-based parallel domain decomposition.

Definition 3 *For cell $u \in \sigma$, the Euclidean distance to a nearest neighbor, denoted by d_u, is defined as:*

$$d_u = \begin{cases} 0 \text{ , if } u \text{ is unoccupied} \\ 0 \text{ , if all other cells are unoccupied} \\ \text{distance to the nearest neighbor, otherwise} \end{cases}$$

A point may have multiple nearest neighbors. We wish to capture the farthest distance along the SFC-array to a nearest neighbor.

Definition 4 *For cell $u \in \sigma$, define X_u as:*

$$X_u = \begin{cases} 0 \text{ , if } u \text{ is unoccupied} \\ 0 \text{ , if all other cells are unoccupied} \\ \text{maximum SFC-array distance to a nearest} \\ \qquad\qquad \text{neighbor, otherwise} \end{cases}$$

Definition 5 *Let Z denote the average of the maximum SFC-array distance to a nearest neighbor, where the average is taken over all cells.*

$$Z = \frac{1}{pm} \sum_{u \in \sigma} X_u$$

Note that mp is the expected number of points in $\sigma(m, p)$. By linearity of expectation, we have

$$\mathbf{E}[Z] = \frac{1}{pm} \sum_{u \in \sigma} \mathbf{E}[X_u] \tag{1}$$

Using the definition of X_u:

$$\begin{aligned} \mathbf{E}[X_u] &= \Pr\{u \text{ is occupied }\} \cdot \Pr\{u \text{ is not the only occupied cell }\} \cdot \mathbf{E}[X_u | X_u \neq 0] \\ &\leq p \sum_{1 \leq d \leq d_{max}} \Pr\{d_u = d\} \cdot \mathbf{E}[X_u \mid d_u = d] \end{aligned} \tag{2}$$

where d_{max} is the largest value that d can take given a $2^k \times 2^k \times 2^k$ block of cells. Combining Eqn (1) and Eqn (2), we have:

$$\mathbf{E}[Z] = \frac{1}{m} \sum_{u} \sum_{d} \Pr\{d_u = d\} \mathbf{E}[X_u \mid d_u = d] \tag{3}$$

where we adopt the notation used for the rest of the paper in which the summation of u implies that it is over all $u \in \sigma$ and that of d implies that it is over all $1 \leq d \leq d_{max}$ unless otherwise specified.

Lemma 3 $\mathbf{E}[X_u | d_u = d] \leq p|A(u, d)|$

Proof Let Y_{ud} be a random variable which denotes the length of the SFC-subarray that contains all occupied cells in $A(u, d)$. If $d_u = d$, then $X_u \leq Y_{ud}$, since both u and its nearest neighbor lie in an SFC-subarray of length Y_{ud}. Thus, it follows that $\mathbf{E}[X_u | d_u = d] \leq \mathbf{E}[Y_{ud}]$. We know that $\mathbf{E}[Y_{ud}] = p|A(u, d)|$, since every cell in $A(u, d)$ is occupied with probability p. ∎

Lemma 4 *For any $u \in \sigma(m, p)$ and $d \in [1, d_{max}]$,* $\Pr\{d_u = d\} \leq (1 - p)^{d^3/8}$

Proof The event $d_u = d$ implies that $C(u, d)$ has no occupied cells. It follows from Lemma 1 that at least $d^3/8$ cells must be unoccupied. The probability of this event is no more than $(1 - p)^{d^3/8}$. ∎

Lemma 5

$$\sum_{1 \leq d \leq d_{max}} d(1 - p)^{\frac{d^3}{8}} \leq \frac{16}{3p}$$

Proof

$$\begin{aligned} \text{Let } S &= \sum_{1 \leq d \leq d_{max}} d(1 - p)^{\frac{d^3}{8}} \\ &\leq \sum_{1 \leq d \leq d_{max}} d e^{\frac{-pd^3}{8}} \end{aligned}$$

Note that d is not necessarily an integer. However, d^2 is an integer because $d^2 = x^2 + y^2 + z^2$ for integers x, y and z. Letting $r = d^2$,

$$\begin{aligned} S &\leq \sum_{r=1}^{\infty} \sqrt{r} e^{\frac{-pr\sqrt{r}}{8}} \\ &\leq \int_0^{\infty} \sqrt{r} e^{\frac{-pr\sqrt{r}}{8}} dr \end{aligned}$$

Let $y = p^{\frac{1}{3}}\sqrt{r}/2$. Then, $dy = p^{\frac{1}{3}} dr/4\sqrt{r}$.

$$\begin{aligned} S &\leq \frac{16}{p} \int_0^{\infty} y^2 e^{-y^3} dy \\ &= \frac{16}{p} \frac{\Gamma(1)}{3} = \frac{16}{3p} \end{aligned}$$

∎

Theorem 6 $\mathbf{E}[Z] = O(n^{2/3})$.

Proof Using Eqn (3), Lemma 2, Lemma 3 and Lemma 4, we get:

$$\begin{aligned} \mathbf{E}[Z] &\leq p^{\frac{1}{3}} n^{\frac{2}{3}} \sum_{1 \leq d \leq d_{max}} d(1-p)^{\frac{d^3}{8}} \\ &\leq \frac{p^{-\frac{2}{3}}}{3} n^{\frac{2}{3}} = O\left(n^{\frac{2}{3}}\right) \end{aligned}$$ ∎

Discussion: Although not shown here, we have generalized this result to show that the nearest neighbor of a point in d dimensions is located at an expected SFC-array distance of $O\left(n^{\frac{d-1}{d}}\right)$. It is not surprising that this measure of locality significantly deteriorates with increasing dimensionality. As the umber of neighboring cells grows exponentially, capturing locality with any one dimensional ordering becomes increasingly difficult. On the surface of it, the three dimensional result indicates that having fewer than $O\left(n^{2/3}\right)$ points per processor might result in all remote accesses, as not even the expected distance of a nearest neighbor falls within the same processor. If this were true, the number of processors would be restricted to $O(n^{1/3})$. Fortunately, this is not the case. Even though the expected distance is large, it turns out that most points have their nearest neighbors much closer. In fact, it will be shown that the number of remote access remains sublinear unless $P = \Theta(n)$.

5. Nearest Neighbor Queries

In Sections 5 and 6, we analyze SFC-based parallel domain decomposition with respect to nearest neighbor and spherical region queries. As mentioned before, the SFC-array is partitioned across processors using block decomposition. We are interested in the expected number of points whose query regions contain points that lie on a remote processor.

Definition 6 *For $u \in \sigma$, the random variable $\mathcal{N}_u$ measures if any nearest neighbor of u is remote:*

$$\mathcal{N}_u = \begin{cases} 0 \text{ , if cell } u \text{ is unoccupied} \\ 0 \text{ , if all NNs of } u \text{ are available locally} \\ 1 \text{ , otherwise} \end{cases} \quad (4)$$

Definition 7 *Let $\mathcal{N}$ be the number of points that have at least one nearest neighbor on a remote processor.*

$$\mathcal{N} = \sum_{u \in \sigma} \mathcal{N}_u$$

Intuitively, $\mathcal{N}$ corresponds to the number of input points that will result in interprocessor communication during a parallel nearest neighbor computation. We wish to compute $\mathbf{E}[\mathcal{N}]$.

Definition 8 *For any point $u \in \sigma(m,p)$, and integer Δ, the Δ-SFC-neighborhood of u is defined as the set of all points whose SFC-array distance from u is less than or equal to Δ.*

In the remainder of this section, we will compute the expectation of $\mathcal{N}$ under the condition $|\sigma(m,p)| > mp/2$. Using the fact $\mathbf{E}[|\sigma(m,p)|] = mp$ and applying a Chernoff bound, we get $\Pr\{|\sigma(m,p)| \leq mp/2\} = O\left(e^{-mp/8}\right)$. Since this condition is true with very high probability, it can be easily seen through conditional probabilities that assuming this condition to be true will not change the asymptotic value of $\mathbf{E}[\mathcal{N}]$.

Theorem 7 *For $P = n^{\alpha}$,* $\mathbf{E}[\mathcal{N}] = O(n^{(3/4+\alpha/4)})$.

Proof Let δ_1, δ_2 be parameters such that $\delta_1 \delta_2 = n^{1-\alpha}$. Let T_u denote the event that $\mathcal{N}_u = 1$, given that u is occupied. Let W_u denote the event that the δ_1-neighborhood of u is not available locally, again under the condition that u is occupied.

$$\begin{aligned} \mathbf{E}[\mathcal{N}_u] &= \Pr\{\mathcal{N}_u = 1\} = p \Pr\{T_u\} \\ &\leq p\left\{\Pr\{W_u\} + \Pr\{T_u | \bar{W}_u\}\right\} \end{aligned}$$

By linearity of expectation:

$$\begin{aligned} \mathbf{E}[\mathcal{N}] &= \sum_{u \in \sigma} \mathbf{E}[\mathcal{N}_u] \\ &\leq p \sum_{u \in \sigma} \Pr\{W_u\} + p \sum_{u \in \sigma} \Pr\{T_u | \bar{W}_u\} \quad (5) \end{aligned}$$

The number of points per processor is at least $\frac{mp}{2P} = \frac{n^{1-\alpha}}{2}$. Among all points assigned to a processor, except for the set of $2\delta_1$ points that are at the extreme left and right ends of the subarray of the SFC that is assigned to the processor, the rest of the points have their δ_1-SFC-neighborhood available locally. Thus, no more than $\frac{2\delta_1}{1/2n^{1-\alpha}} = \frac{4}{\delta_2}$ fraction of the points u have $W_u = 1$. Using this, and the fact $mp = n$ in Equation 5:

$$\mathbf{E}[\mathcal{N}] \leq \frac{4n}{\delta_2} + p\sum_{u\in\sigma} \Pr\{T_u|\bar{W}_u\} \quad (6)$$

Consider a filled cell u. For any $d > 0$, all points that are at a distance less than or equal to d are contained in the standard cube $A(u,d)$. All occupied cells in $A(u,d)$ lie in a contiguous portion of the SFC array. The number of occupied cells in $A(u,d)$ is the binomial random variable $\mathcal{B}(|A(u,d)|,p)$ i.e. the number of heads in $|A(u,d)|$ coin tosses, where the probability of a head on each toss is p. Recall that random variable d_u is the physical distance from u to its nearest neighbor. If the nearest neighbor of u is remote, and the δ_1-SFC-neighborhood of u is local, then it must be true that $A(u,d_u)$ has more than δ_1 cells.

Thus,

$$\Pr\{T_u|\bar{W}_u\} \leq \sum_d \Pr\{d_u = d\} \times \Pr\{\mathcal{B}(|A(u,d)|,p) > \delta_1\}$$

$$\sum_{u\in\sigma}\Pr\{T_u|\bar{W}_u\} \leq \sum_{u\in\sigma}\sum_d \Pr\{d_u = d\} \times \Pr\{\mathcal{B}(|A(u,d)|,p) > \delta_1\} \quad (7)$$

Using Lemma 4 and Equation 7 in Equation 6, $\mathbf{E}[\mathcal{N}]$

$$\leq \frac{4n}{\delta_2} + p\sum_{u\in\sigma}\sum_d (1-p)^{\frac{d^3}{8}} \cdot \Pr\{\mathcal{B}(|A(u,d)|,p) > \delta_1\}$$

$$= \frac{4n}{\delta_2} + p\sum_d (1-p)^{\frac{d^3}{8}} \cdot \sum_{u\in\sigma}\Pr\{\mathcal{B}(|A(u,d)|,p) > \delta_1\} \quad (8)$$

Using Lemmas 8 and 5 in Equation 8, we get:

$$\begin{aligned}\mathbf{E}[\mathcal{N}] &\leq \frac{4n}{\delta_2} + p\sum_d (1-p)^{d^3/8} 13md\frac{p^{1/3}}{{\delta_1}^{1/3}} \\ &= \frac{4n}{\delta_2} + \frac{13np^{1/3}}{{\delta_1}^{1/3}}\sum_d d(1-p)^{d^3/8} \\ &\leq \frac{4n}{\delta_2} + \frac{208n}{3{\delta_1}^{1/3}p^{2/3}}\end{aligned}$$

Suppose $\delta_1 = n^\epsilon$, so that $\delta_2 = n^{1-\alpha-\epsilon}$. Rewriting the above expression in terms of ϵ:

$$\begin{aligned}\mathbf{E}[\mathcal{N}] &\leq \frac{4n}{n^{1-\alpha-\epsilon}} + \Theta(\frac{n}{n^{\epsilon/3}}) \\ &= 4n^{\alpha+\epsilon} + \Theta(n^{1-\epsilon/3})\end{aligned}$$

Minimizing the above expression for $\mathbf{E}[\mathcal{N}]$ with respect to ϵ, we finally get $\mathbf{E}[\mathcal{N}] = O(n^{3/4+\alpha/4})$. ∎

We now prove Lemma 8 that is used in proving Theorem 7. Implications of the result of Theorem 7 are deferred to Section 7.

Lemma 8 *For $r > 0$ and $\delta > \log k$,*

$$\sum_{u\in\sigma}\Pr\{\mathcal{B}(|A(u,r)|,p) > \delta\} \leq 13mr\frac{p^{1/3}}{\delta^{1/3}}$$

Proof We partition σ into $k+1$ sets, $\sigma_0, \sigma_1, \ldots, \sigma_k$ as follows. For each cell $u \in \sigma_i$, $|A(u,r)| = 8^{k-i}$. From Lemma 2, we know $|\sigma_i| \leq 6r \cdot 4^k \cdot 2^i$. Let "$LHS$" denote the left hand side of the expression in the lemma.

$$\begin{aligned}LHS &= \sum_{i=0}^{k}\sum_{u\in\sigma_i}\Pr\{\mathcal{B}(|A(u,r)|,p) > \delta\} \\ &= \sum_{i=0}^{k} 6r4^k2^i \Pr\{\mathcal{B}(8^{k-i},p) > \delta\} \\ &= 6r4^k\sum_{i=0}^{k} 2^i \Pr\{\mathcal{B}(8^{k-i},p) > \delta\}\end{aligned}$$

Let i^* be the smallest integer such that $8^{k-i^*}p < \delta/6$. Thus $2^{i^*} \leq 2^{k+1}\left(\frac{6p}{\delta}\right)^{1/3}$. We can breakdown the above sum as follows.

$$\begin{aligned}LHS &= 6r4^k\sum_{i=0}^{i^*} 2^i + 6r4^k\sum_{i=i^*+1}^{k}\Pr\{\mathcal{B}(8^{k-i},p) > \delta\} \\ &\leq 6r4^k2^{i^*+1} + 6r4^k\sum_{i=i^*+1}^{k} 2^{-\delta} \\ &\leq 12r4^k2^{i^*} + 6r4^kk2^{-\delta}\end{aligned}$$

We have used the following Chernoff bound: if X is a binomial random variable whose expectation is μ, then for any $\gamma > 6\mu$, $\Pr\{X \geq \gamma\} < 2^{-\gamma}$. Since $\delta > \log k$, the above expression is no more than $13r4^k2^{i^*} = 13r8^k\frac{p^{1/3}}{\delta^{1/3}} = 13mr\frac{p^{1/3}}{\delta^{1/3}}$. ∎

6. Spherical Region Queries

In molecular dynamics simulations, the Lennard-Jones potential between atoms u and v is computed iff their physical separation $d(u,v) \leq r$, where r is a user provided cutoff distance. Therefore, it is desirable that such pairs of atoms reside on the same processor to minimize communication overhead. In this section, we analyze the total number of points requiring remote accesses for such a spherical region query. We assume that the cutoff radius r is a constant that is independent of m, p or P.

Definition 9 *For a point $u \in \sigma(m,p)$, the ϕ-neighborhood of u is defined as the set $\{v \in \sigma(m,p) | d(u,v) \leq \phi\}$.*

Definition 10 *For $u \in \sigma$, the random variable R_u corresponds to whether the point in u has remote interactions or not, and is defined as:*

$$R_u = \begin{cases} 0 \text{ , if cell } u \text{ is unoccupied} \\ 0 \text{ , if } r\text{-neighborhood of } u \text{ is local} \\ 1 \text{ , otherwise} \end{cases} \quad (9)$$

Definition 11 *The random variable R which corresponds to the fraction of points that have remote interactions is defined as:*

$$R = \frac{1}{mp} \sum_{u \in \sigma} R_u \quad (10)$$

Clearly, smaller the value of R, lesser the number of remote interactions and greater the efficiency of computation. Our goal is to calculate $\mathbf{E}[R]$. As in Section 5, in the remainder of this section, $\mathbf{E}[R]$ is computed subject to the condition $|\sigma(m,p)| > mp/2$.

Theorem 9 *If the number of processors (P) is n^α, then $\mathbf{E}[R] = O(n^{3/4+\alpha/4})$.*

Proof The proof is along similar lines to the proof of the parallel nearest neighbor. Let δ_1, δ_2 be parameters such that $\delta_1 \delta_2 = n^{1-\alpha}$. Let V_u denote the event that $R_u = 1$, given that u is occupied. Let W_u denote the event that the δ_1-neighborhood of u is not available locally, again under the condition that u is occupied.

$$\mathbf{E}[R_u] \leq p\left\{\Pr\{W_u\} + \Pr\{V_u | \bar{W}_u\}\right\}$$

Using a similar argument as in the proof of Theorem 7 to bound $\Pr\{W_u\}$, and applying linearity of expectation, we get:

$$\mathbf{E}[R] = \frac{4n}{\delta_2} + p \sum_{u \in \sigma} \Pr\{V_u | \bar{W}_u\} \quad (11)$$

Consider a filled cell u. All points that are in a r-neighborhood of u are contained in the standard cube $A(u,r)$. If the r-neighborhood of u is not fully local, and the δ_1-SFC-neighborhood of u is local, then it must be true that $A(u,r)$ has more than δ_1 cells.

$$\begin{aligned} \sum_{u \in \sigma} \Pr\{V_u | \bar{W}_u\} &\leq \sum_{u \in \sigma} \Pr\{\mathcal{B}(|A(u,r)|, p) > \delta_1\} \\ &\leq 13mr \frac{p^{1/3}}{{\delta_1}^{1/3}} \quad (12) \end{aligned}$$

where we have used Lemma 8. Using Equation 12 in Equation 11, and using $mp = n$:

$$\mathbf{E}[R] \leq \frac{4n}{\delta_2} + 13nr \frac{p^{1/3}}{{\delta_1}^{1/3}}$$

If r is a constant, then, using an argument similar to the one in Section 5, we can minimize the above expression under the constraint $\delta_1 \delta_2 = n^{1-\alpha}$ to get $\mathbf{E}[R] = O(n^{3/4+\alpha/4})$. ∎

7. Discussion

As shown in Sections 5 and 6, the total number of points requiring remote accesses grows as $O\left(n^{3/4+\alpha/4}\right)$ for both nearest neighbor and spherical region queries. A number of interesting conclusions can be drawn from this result. The number of remote accesses starts out at $O\left(n^{3/4}\right)$ for negligible α, and grows to $O(n)$ as α reaches 1 (for $P = n$ processors). The number of remote accesses grows at least as $n^{3/4}$, no matter how small the number of processors used. Thus, interprocess communication overhead does not significantly improve by reducing the number of processors.

On the other hand, the total number of remote accesses grows sublinearly with the problem size n, as long as the number of processors used is sublinear in n. This implies that the computational complexity is greater than the communication complexity. As the number of processors P is increased, the rate at which total remote accesses grow scales only as $\sqrt[4]{P}$. These results favor the use of large parallel systems.

Once the points within a query region are found, there is typically $O(1)$ computation per point. Even assuming that there are only a constant number of points in each query region, the total computational complexity grows as $O(n)$. Thus, the ratio of total computation cost to total communication cost is given by $O\left(n^{(1-\alpha)/4}\right)$. This ratio increases with increasing n, thus improving the situation as n increases. A critical issue for a parallel algorithm to be practically useful is the ability to limit the communication overhead in relation to the computational costs. As the ratio of computational complexity to communication complexity is an increasing function of n, a lower percentage of time spent in communication can be achieved for a fixed number of processors by increasing n. These analyses demonstrate that SFCs are useful for domain decomposition on large parallel systems.

8. Conclusions and Open Problems

In this paper, we presented a formal analysis of the effectiveness of SFCs for parallel domain decomposition in

the context of nearest neighbor and spherical region queries. Our analysis provides new and valuable insights, and raises several important questions for further investigation. Our results are generic and apply for any space filling curve. Several researchers have reported that Hilbert SFCs are better at preserving locality than, say, the Morton ordering or Z-SFC [8]. It would be interesting to see if an SFC-specific analysis can be incorporated in our framework to gain further insights. It would be especially important to find if the different SFCs are equivalent in their locality properties (up to a constant factor) or if there are more significant differences. Analysis of SFCs for more complex, and non-uniform distributions also remains an open problem.

References

[1] S. Aluru and F. Sevilgen, Parallel domain decomposition and load balancing using space-filling curves,*Proc. 4th IEEE International Conference on High Performance Computing*, (1997) 230-235.

[2] C. Gotsman and M. Lindenbaum, The metric properties of discrete space-filling curves, *IEEE Transactions on Image Processing, 5(5)* (1996) 794-797.

[3] M. Griebel and G. Zumbusch, Hash based adaptive parallel multilevel methods with space filling curves, *Proc. NIC Symposium* (2001), In H. Rollnik and D. Wolf, Editors, John von Neumann Institute for Computing (NIC) Series, 479-492, 2002.

[4] D. Hilbert, Uber die stegie Abbildung einer Linie auf Flachenstuck, *Math. Ann., 38* (1891) 459-460.

[5] V.J. Leung, E.M. Arkin, M.A. Bender, D.P. Bunde, J. Johnston, A. Lal, J.S.B. Mitchell, C.A. Phillips and S.S. Seiden, Processor allocation on Cplant: Achieving general processor locality using one-dimensional allocation strategies, *Proc. IEEE International Conference on Cluster Computing (CLUSTER)* (2002) 296-304.

[6] Shen-Yi Lin, Chih-Shen Chen, Li Liu and Chua-Huang Huang, Tensor Product Formulation for Hilbert Space-Filling Curves, *Proc. International Conference on Parallel Processing* (2003), 99-106.

[7] B. Moon, H.V. Jagadish, C. Faloutsos and J.H. Saltz, Analysis of the clustering properties of Hilbert space-filling curve, *IEEE Transactions on Knowledge and Data Engineering, 13(1)* (2001) 1240-141.

[8] G.M. Morton, A computer oriented geodetic data base and a new technique in file sequencing, IBM, Ottawa, Canada (1966).

[9] A. Nakano, R.K. Kalia and P. Vashishta, Scalable molecular dynamics visualization, and data management algorithms for materials simulations, *IEEE Computing in Science and Engineering, 1(5)* (1999) 39-47.

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

[11] J. Steensland, S. Chandra and M. Parashar, An application-centric characterization of domain-based SFC partitioners for parallel SAMR, *IEEE Transactions on Parallel and Distributed Systems, 13(12)* (2002) 1275-1289.

[12] M.S. Warren and J.K. Salmon, A Parallel Hashed Oct-Tree N-Body Algorithm, *Proc. Supercomputing '93* (1993) 12-21.