Space-Filling Curve Partitioning for Domain Decomposition

H. Carter Edwards
Texas Institute for Computational and Applied Mathematics
http://www.ticam.utexas.edu/

May 1997

The Partitioning Problem

Partition the Domain into Subdomains

- Finite Differences $\Rightarrow$ Mesh Partitioning
- Finite Elements (Volumes) $\Rightarrow$ Element Clustering

Goals

- Load balance subdomains' computational work
- Minimize communication $\Leftrightarrow$ subdomain interfaces
- Efficiency: $\boxed{\text{partitioning cost} << \text{solution cost}}$
  - •• Computational resources (time & space)
  - •• Human resources (complexity)

Space-Filling Curve (SFC)

Mathematics: Cantor (1878), . . ., Hilbert (1891), . . ., Fractals

Applications

- Domain Decomposition Partitioning: Warren and Salmon (1993)

Mathematics, e.g. Hilbert SFC

Recursive Map:

- $S : [0,1] \leftrightarrow [0,1]^n$
- Example, 1^{st} Level:

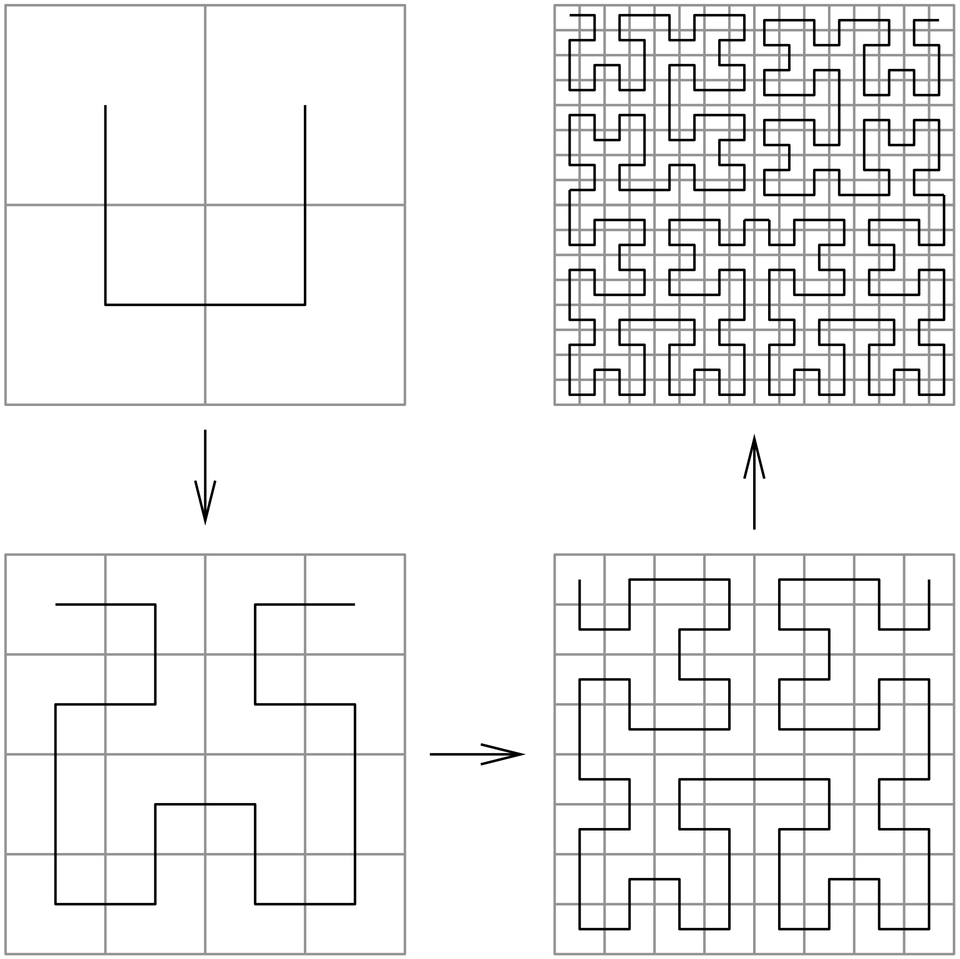
$$[0,\tfrac{1}{4}) \leftrightarrow [0,\tfrac{1}{2}) \times [\tfrac{1}{2},1]$$

$$[\tfrac{1}{4},\tfrac{1}{2}) \leftrightarrow [0,\tfrac{1}{2}) \times [0,\tfrac{1}{2})$$

$$[\tfrac{1}{2},\tfrac{3}{4}) \leftrightarrow [\tfrac{1}{2},1] \times [0,\tfrac{1}{2})$$

$$[\tfrac{3}{4},1] \leftrightarrow [\tfrac{1}{2},1] \times [\tfrac{1}{2},1]$$

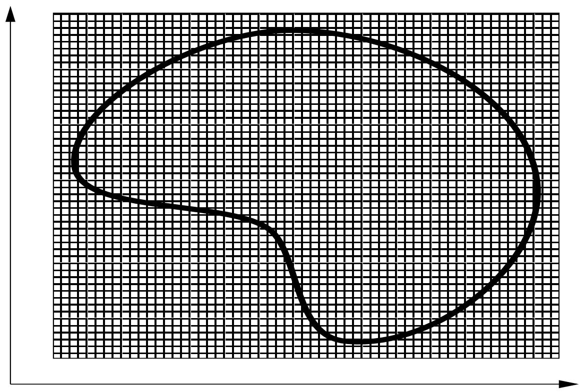
- Locality $[0,1] \rightarrow [0,1]^n$
- Computationally Cheap



Application

Arbitrary Bounded Domain $\Omega \subset \Re^n$

- Define a bounding box $B \supset \bar{\Omega}$
- Define an affine map $T : [0,1]^n \leftrightarrow B$
- $T \circ S : [0,1] \leftrightarrow B \supset \Omega$
- $S^{-1} \circ T^{-1} : \Omega \mapsto [0,1]$



A Multiple Use Tool:

- Partitioning for Domain Decomposition
- Ordering Elements/Nodes for Solvers
- Naming Elements/Nodes for Distributed Data Structures

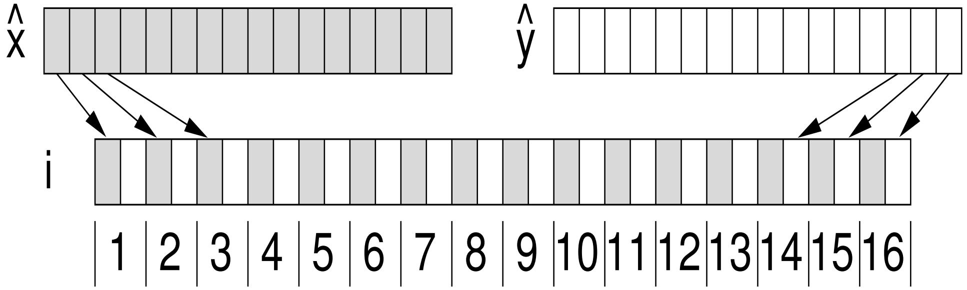
Map Details . . .

Simple Affine Map T

- $\left.\begin{array}{l} x_{\text{inf}} < x < x_{\text{sup}} \\ y_{\text{inf}} < y < y_{\text{sup}} \end{array}\right\} \forall (x, y) \in \Omega$
- $(\hat{x}, \hat{y}) = T^{-1}(x, y) = \left(\dfrac{x - x_{\text{inf}}}{x_{\text{sup}} - x_{\text{inf}}}, \dfrac{y - y_{\text{inf}}}{y_{\text{sup}} - y_{\text{inf}}} \right)$

SFC Map $S^{-1} : [0, 1]^n \mapsto [0, 1]$

- Discretize:



- Sequence of bit operations: gray code, rotations, and reflections

Domain Decomposition Partitioning

$\mathbf{1})$ Associate each element/node with a unique $\Re^n$ coordinate

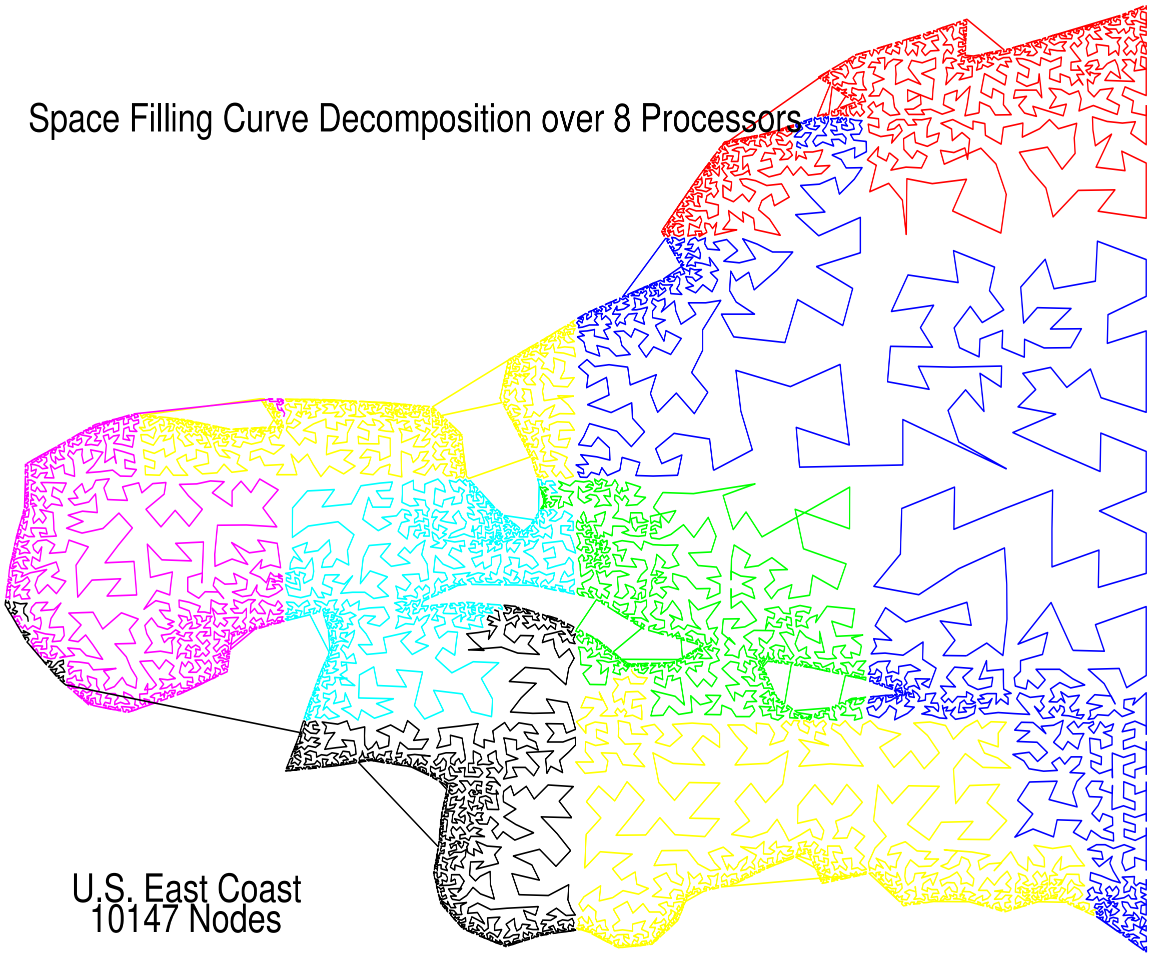
$\mathbf{2})$ Apply SFC $\circ$ Affine Map, $S^{-1} \circ T^{-1} : \Omega \mapsto [0, 1]$

$\Rightarrow$ Linear ordering of elements/nodes

$\mathbf{3})$ Partition for Load Balance

- Assign computational weight to each element/node
- Partition linearly ordered elements/nodes into P balanced subdomains
- Simple/cheap order "N" algorithm
- SFC locality preserving $\rightarrow$ $\sim$minimizes interfaces

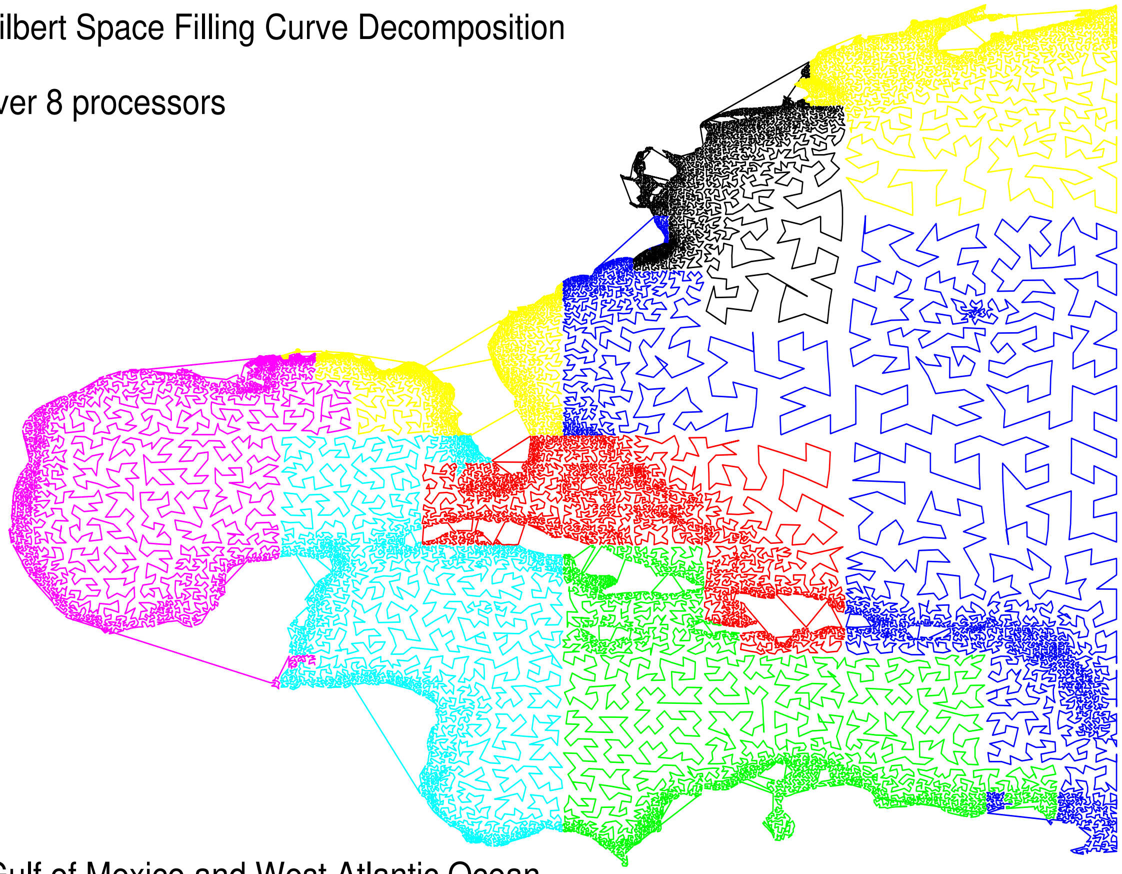
Examples


Space Filling Curve Decomposition over 8 Processors
U.S. East Coast
10147 Nodes

Examples

Hilbert Space Filling Curve Decomposition

over 8 processors



Gulf of Mexico and West Atlantic Ocean

31435 Nodes

Potential Enhancements

Hierarchical Partitioning

- SFC partition elements into dense “meta-elements”
- SFC partition meta-elements into subdomains

Polyalgorithm Partitioning

- SFC partition elements into dense “meta-elements”
- *Inertial bisection* algorithm: meta-elements into subdomains

Ordering Elements/Nodes for Solvers

Locality $[0, 1] \to \Omega$

- Neighboring elements/nodes in $[0, 1]$ are connected in Ω

$\Rightarrow$ Dense matrix diagonals for preconditioning

Sequence of Partitionings

- Substructuring for hierarchical domain decomposition
- “heart of” sparse direct solvers

Naming Elements/Nodes for Distributed Data Structures

Global Distributed Data Set of elements/nodes

- Partitioned among processors
- Element/node connectivity, between processors?

Global Name Space for elements/nodes

- An element/node $\mapsto$ unique point $(x, y) \in \Omega$
- Mapping: $\Omega \ni (x, y) \mapsto i \in [0, 1]$
- i, a unique global name for the element/node

Scalable Distributed Dynamic Array

A parallel infrastructure supporting distributed dynamic data structures, e.g. adaptive meshes.

- Distributed *data set* D, e.g. elements/nodes
- Global *index space* I: indices + linear ordering
- *Mapping* $F : D \mapsto I$

Space-Filling Curve Connection

- Global name space $[0, 1]$ is an *index space*
- Space-filling curve *mapping* data $\leftrightarrow$ indices

Features

Distributed Data Management

- Automatic data distribution
- Caching of shared data
- Data redistribution

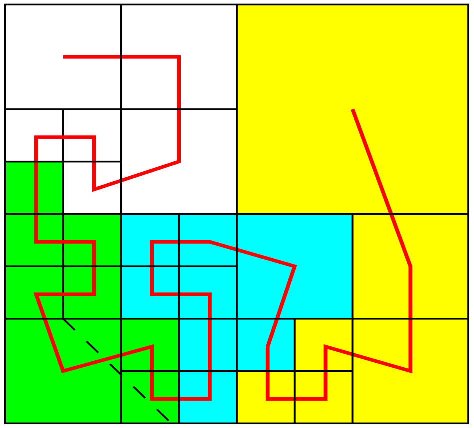
Virtual Shared Memory Interface

- Transparent interprocessor communications
- High level of abstraction: get/put/delete

Implementation

- Message-Passing Interface (MPI) Standard, version 1.1
- C++ Standard Template Library (STL)

Scalable Distributed Dynamic Array

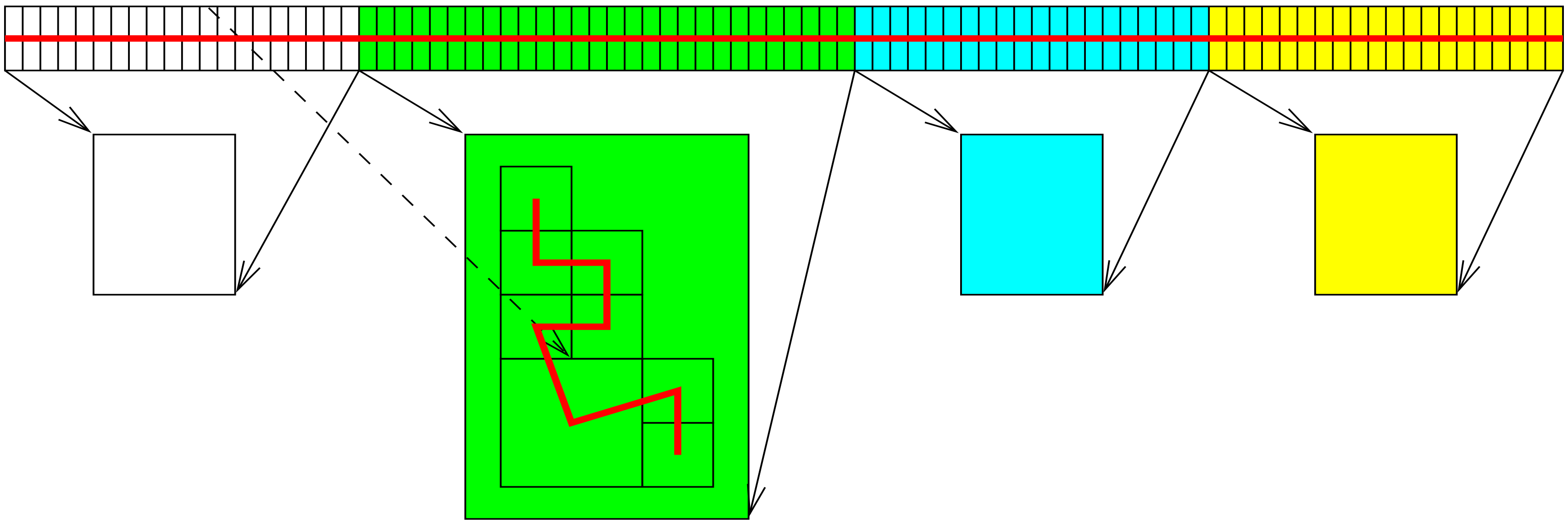


Space-Filling Curve

=> Domain Decomposition Partitioning

=> Index Space Partitioning

=> Data Distribution



Storage Locality <=> Index Locality => Geometric Locality

Conclusion

Space-Filling Curve (SFC), an efficient multipurpose tool

- Domain decomposition partitioning
- Element/node ordering
- Global name space

SFC Partitioning for Domain Decomposition

- Computationally cheap
- “Good” partitioning: subdomain surface-to-volume ratio