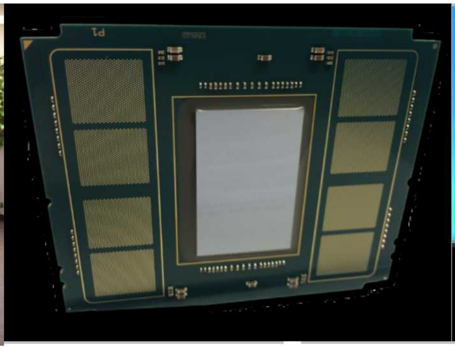
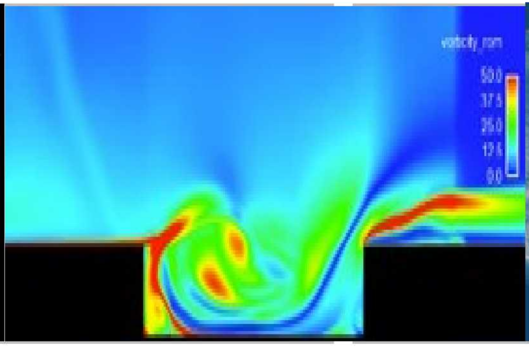
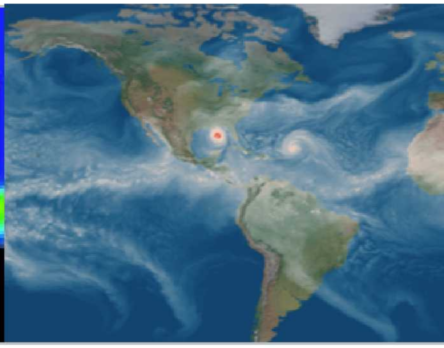
SAND2019-1993C



Partitioning Strategies for Exascale

Erik Boman

SIAM CSE19, Spokane, Feb. 2019

Thanks to:

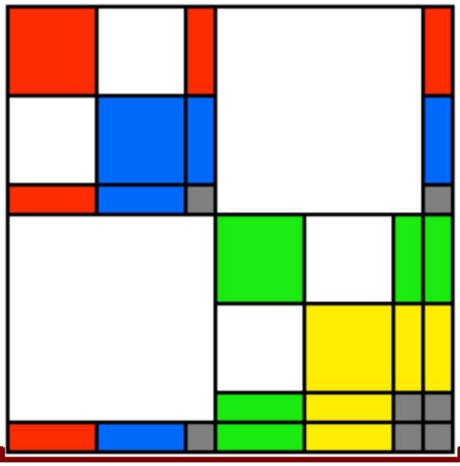
Colleagues:
Seher Acer, Karen Devine, Bruce Hendrickson, Siva Rajamanickam, Michael Wolf

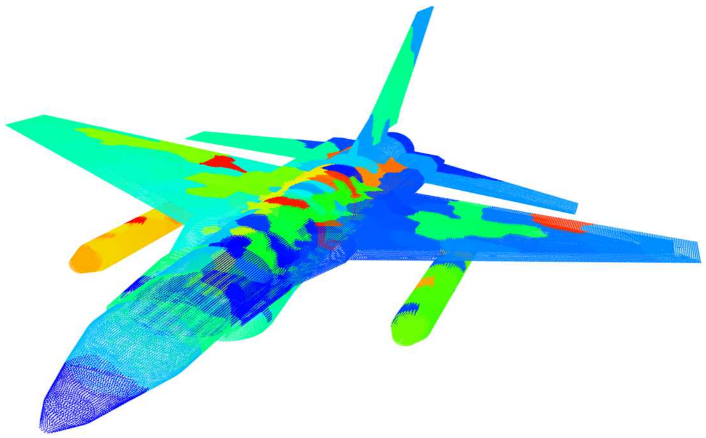
Funding: Exagraph, Exascale Computing Project (ECP), DOE

Partitioning for parallel computing

Problem: Partition the work among processors to balance the load and keep communication low.

- Fundamental problem (distributed memory)
- Importance may grow as HPC becomes increasingly parallel (exascale)
- Might become less important in the future if dynamic task migration systems improve

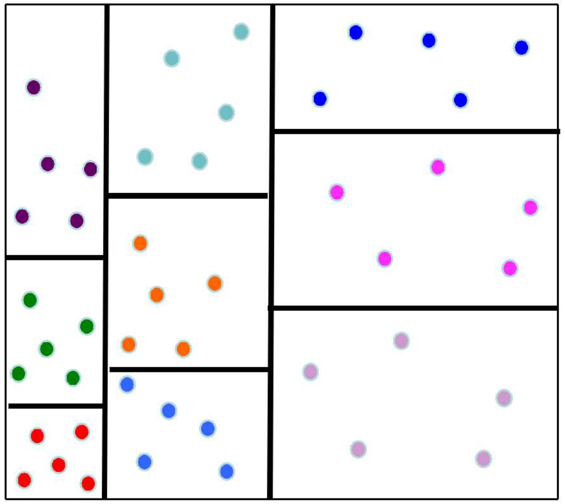


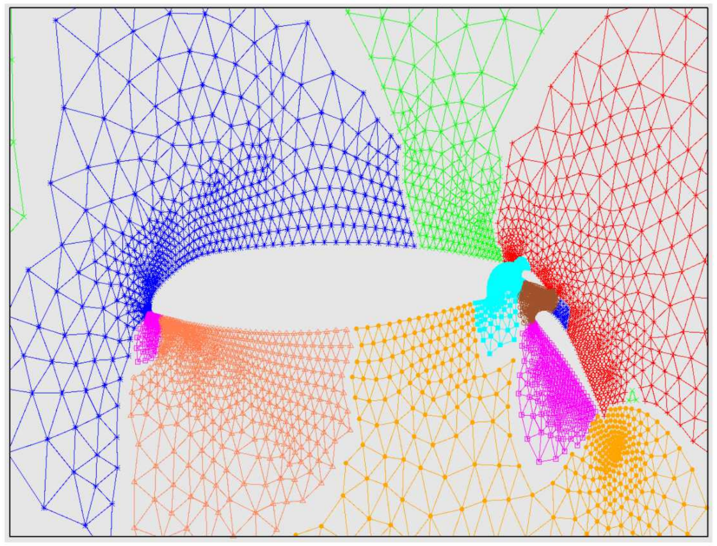


Partitioning Strategies

Use geometry, connectivity, graph/hypergraph model?

- Hypergraphs generalize graphs





	Speed	Edge cut	Comm. Vol.
Geometric	fast	poor	poor
Graph	medium	good	good
Hypergraph	slow	good	best

Graph Partitioning

Problem: Partition the vertices of the graph into *k* roughly equal disjoint sets such as to minimize

a) Cut edges, or
b) Communication volume (boundary vertices)

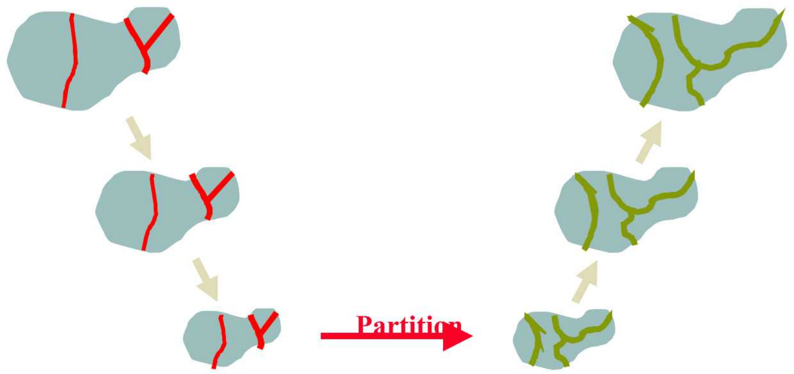
A closely related problem is to find small *vertex separators*.

Algorithms (chronological order):

- Local greedy
  - BFS and level sets
  - Kernighan-Lin and FM
- Spectral
- Multilevel

Multi-Level Graph Partitioning

- Highly successful graph partitioning method
  - Bui & Jones (1993); Hendrickson & Leland (1993); Karypis and Kumar (1995)
  - Construct smaller approximations to graph.
  - Perform graph partitioning on coarse graph.
  - Propagate partition back, refine as needed (typically each level)
- Software:
  - Graphs: (Par-)Metis, Scotch, KaHip/Kaffpa, …
  - Hypergraphs: PaToH, hMetis, Zoltan/PHG, Mondriaan



Exascale challenges:

1. Parallel quality.
   - We want high quality independent of number of processors. Currently only geometric methods are scalable in this sense, not multilevel graph partitioners.

2. Multi-GPU.
   - Current partitioners run on CPU only. Need GPU and multi-GPU code for current and future supercomputers.

3. Other objectives.

Exascale: My thoughts

Multilevel:

- Most popular method today (Metis, Scotch, KaHiP, etc.)
- Hard to parallelize due to (KL/FM) refinement
- Poorly suited for GPU, need another approach!

Spectral:

- Based on linear algebra, so well suited for GPU.
- Worse quality, but not by much so still feasible
- Leverage investments in eigensolvers (e.g., LOBPCG)
- May need to combine with multilevel?

Spectral partitioning

- Use eigenvectors of the graph Laplacian: *L(G) = D-A*
  - Partitioning: Fiedler ('73), Donath, Hoffman ('73), Pothen, Simon, Liou ('90)
  - Clustering: Hagen, Kahng ('92), Shi, Malik ('00), Ng et al. ('02)

- Algorithm outline
  - Compute $min \frac{x^T L(G) x}{x^T x}$ using eigensolver
  - Bisect: $V_0 = \{x_i \text{ s.t. } x_i < \text{median}(x)\}$, $V1 = \{x_i \text{ s.t. } x_i >= \text{median}(x)\}$
  - Recurse on subgraphs

- Variations:
  - Spectral quadrisection and octasection using 2-3 eigenvectors
  - Use several eigenvectors for low-dimensional embedding, then geometric clustering or partitioning
- Why does it work?
  - Continuous relaxation of discrete (binary) optimization problem
  - Discrete bisection method models graph edge cut *exactly*

Spectral partitioning: Current work

We are working on parallel code for ECP (ExaGraph).

- Spectral partitioning using Trilinos/Anasazi
- Will run on multi-GPU (soon)
- Evaluating several algorithmic options
- Will deliver in Zoltan2 (Trilinos)
- May also be used within multilevel framework (future)

Related effort by Pieter Ghysels (LBL) for vertex separators and nested dissection ordering.

History lesson: "Graph Partitioning - Has the Emperor No Clothes?", Hendrickson, '98

- Bruce Hendrickson gave a talk at Irregular'98 pointing out several deficiencies and challenges in graph partitioning.
- This was so popular it was recycled/improved many times
- Status 20 years later:
  - Only a few issues have been resolved
  - Most are still there!

Introduction

- *Two happy occurrences.*
  - *(1) Good graph partitioning tools & software.*
  - *(2) Good parallel efficiencies for many applications.*
- *Is the latter due to the former?*
- *Yes, but also no.*
- *We have been lucky!*
  - *Graph partitioning optimizes wrong objectives.*
  - *Model is insufficiently general.*
  - *Software tools often poorly designed.*

Flaw 1: The Edge Cut Deceit

- *Generally believed that*
  - *"Edge Cuts = Communication Cost".*

- *This belief is behind the use of graph partitioning.*

- *In reality:*
  - *Edge cuts are not equal to communication volume.*
  - *Communication volume not equal to communication cost.*

Flaw 2: Lack of Expressibility

- *Implicit assumptions of graph model*
  - *Edge cuts = communication volume.*
  - *Single vertex weight can encode computational cost.*
  - *Computational cost known a priori.*
  - *Data dependencies are bi-directional.*
- *Assumptions often invalid!*
  - *Computation consists of multiple phases.*
  - *Work depends on decomposition.*
    - *E.g. sparse factorization on each subdomain.*
  - *Dependencies are directed*
    - *e.g. unsymmetric or rectangular matvec.*

Update: Incorrect/inexact metrics

Hypergraph model correctly models communication volume (SpMV).

- Software available:
  - PaToH, hMetis, Mondriaan (serial)
  - Zoltan/PHG (parallel)

Still less widely used than graph partitioners. Why?

- Little difference for meshes and low-degree graphs
- Often slower
- Hypergraphs obscure concept, only known by experts

Update: Communication metrics

- Hypergraphs do communication volume, but that only approximates execution time.
- Not single metric is ”the right one”:
  - Total comm. volume
  - Max volume (per proc.)
  - #send/recv messages
- Recent papers address this:
  - “Encapsulating multiple communication metrics”, B. Ucar and C. Aykanat, SISC, 2004
  - “Improving performance of sparse matrix dense matrix multiplication on large-scale parallel systems”, S. Acer, O. Selvitopi, C. Aykanat, Parallel Computing, 2016
  - “A recursive hypergraph bipartitioning framework for reducing bandwidth and latency costs simultaneously”, O. Selvitopi, S. Acer, C. Aykanat, IEEE TPDS, 2017

Update: 2D or nonzero partitioning

- To minimize number of messages, use 2D matrix partitioning
    - This implies we partition the *nonzeros* of the matrix
- A checkerboard (2D block) partitioning has at most 2*sqrt(p) messages
    - Dense: tight, sparse: upper bound
- Several recent algorithms:
    - Coarse-grain (checkerboard) using hypergraphs
        - Catalyurek & Aykanat
    - Fine-grain, large hypergraph
        - Catalyurek & Aykanat
    - Mondriaan
        - Vastenhouw & Bisseling
    - Medium-grain
        - Pelt & Bisseling
    - 2D using 1D graph/hypergraph
        - B., Devine, Rajamanickam

Update: Multiple weights/constraints

- Important in multi-physics
  - Ex: Balance both particles and mesh
  - Ex: Balance both computation and memory
- Makes partitioning problem even harder!
- Partially supported in some partitioners
  - Metis/ParMetis, PaToH, Zoltan
- Still not widely used
  - Predict higher usage in the future

Conclusions

- Partitioning has played a key role in the success of (distributed memory) parallel computing
- Graph partitioning has some flaws but is still widely used
- Hypergraph partitioning and extensions have some advantages and slowly gaining traction
- Many challenges from ‘90s still remain!
- Exascale computing poses new challenges
  - Sandia Exagraph team is working on it

- Recent problems/work in data science (network science) on clustering and community detection is related but different