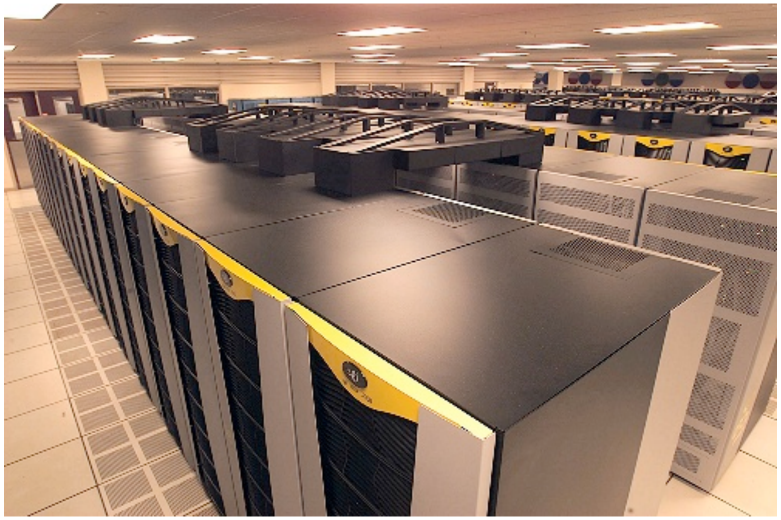
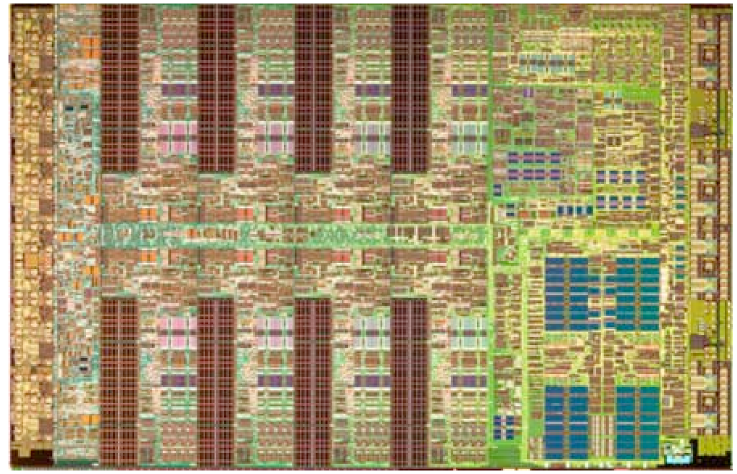
An Interactive Environment for Combinatorial Scientific Computing

Viral B. Shah
John R. Gilbert
Steve Reinhardt

With thanks to: Brad McRae, Stefan Karpinski, Vikram Aggarwal, Min Roh

HPC today is exciting !





INTERACTIVE supercomputing

UCSB

Complex software stack

Computational ecology, CFD, data exploration

Applications

CG, BiCGStab, etc. + combinatorial preconditioners (AMG, Vaidya)

Preconditioned Iterative Methods

Graph querying & manipulation, connectivity, spanning trees,
geometric partitioning, nested dissection, NNMF, . . .

Graph Analysis & PD Toolbox

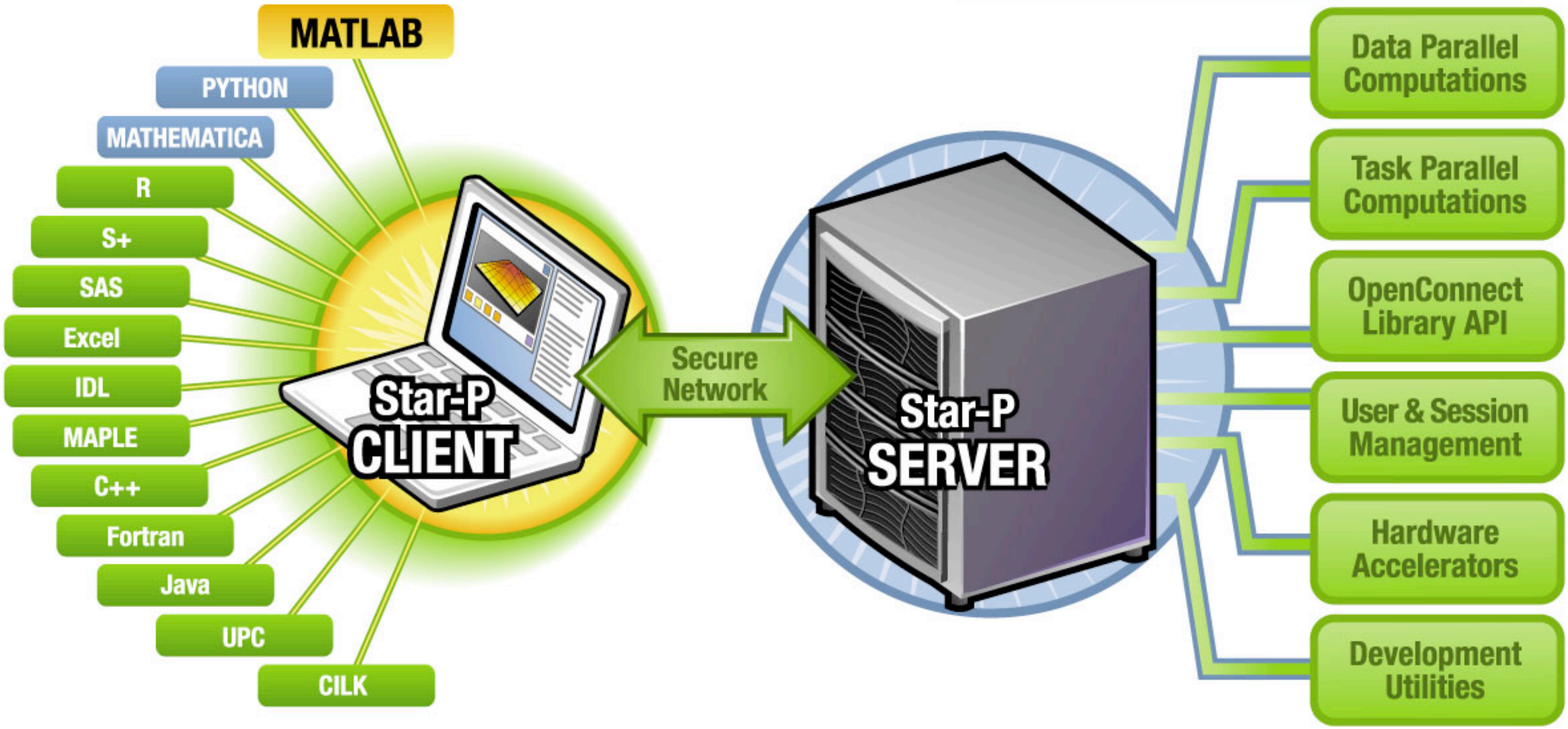
Arithmetic, matrix multiplication, indexing, solvers (\, eigs)

Distributed Sparse Matrices

Star-P

```
A = rand(4000*p, 4000*p);
x = randn(4000*p, 1);
y = zeros(size(x));
while norm(x-y) / norm(x) > 1e-11
    y = x;
    x = A*x;
    x = x / norm(x);
end;
```

Star-P architecture



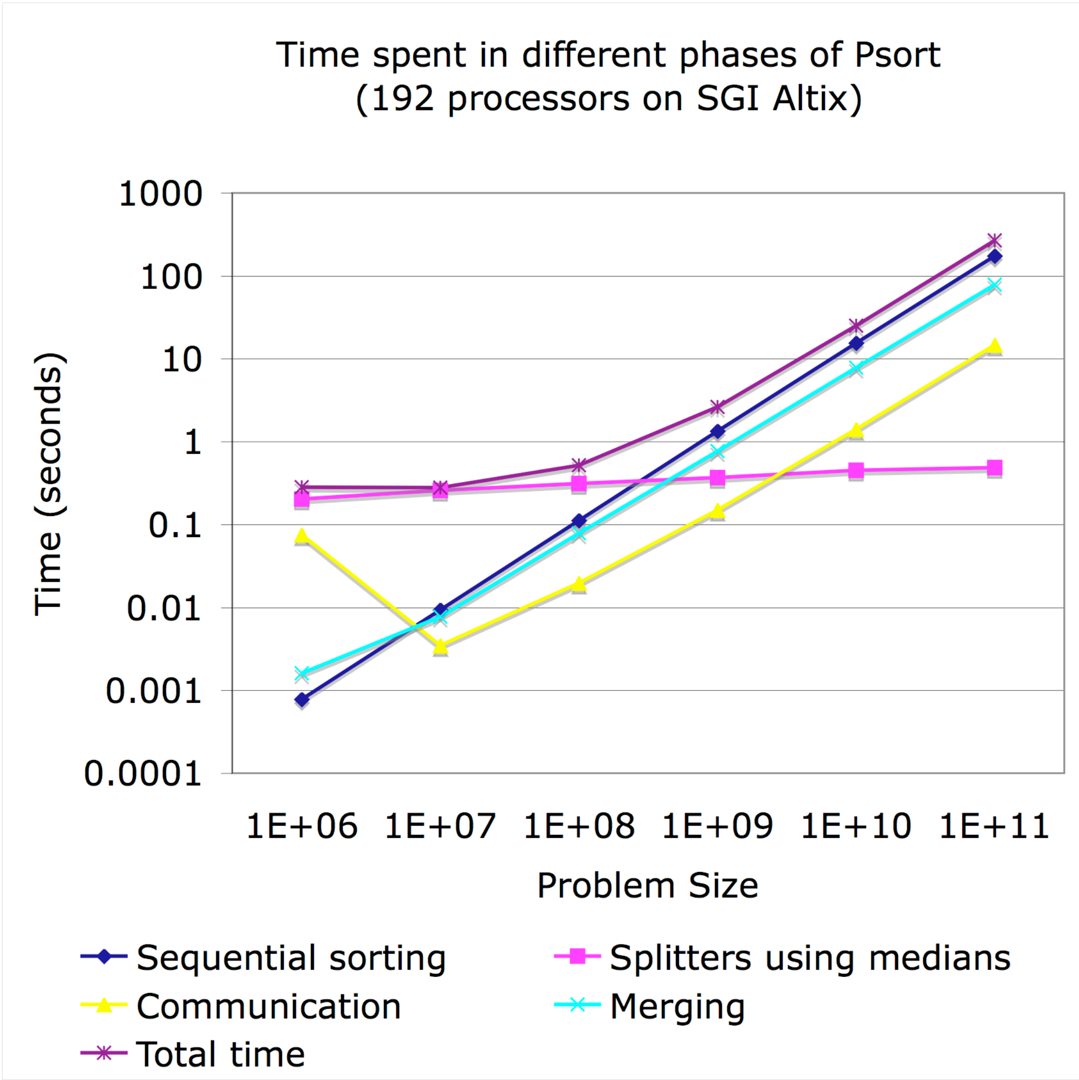
Parallel sorting

3	6	8	1	5	4	7	2	9

1	2	3	4	5	6	7	8	9

- Simple, widely used combinatorial primitive
- `[V, perm] = sort (V)`
- Used in many sparse matrix and array algorithms: sparse(), indexing, concatenation, transpose, reshape, repmat etc.
- Communication efficient

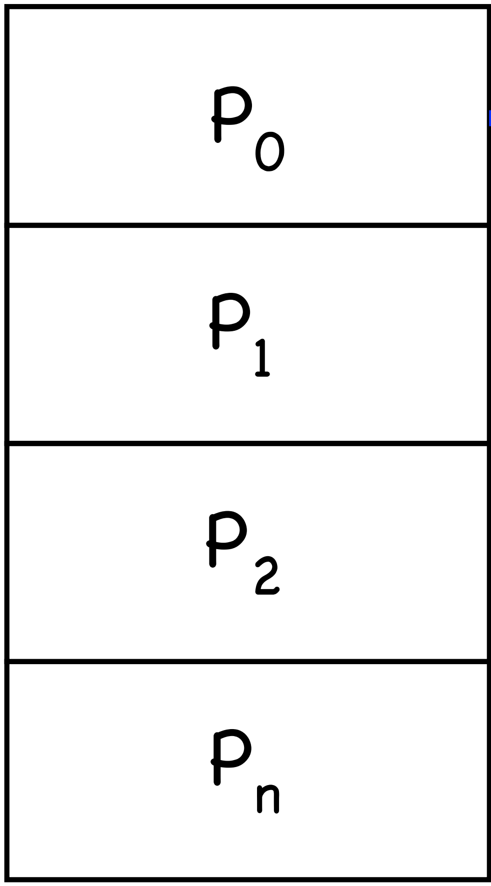
Sorting performance

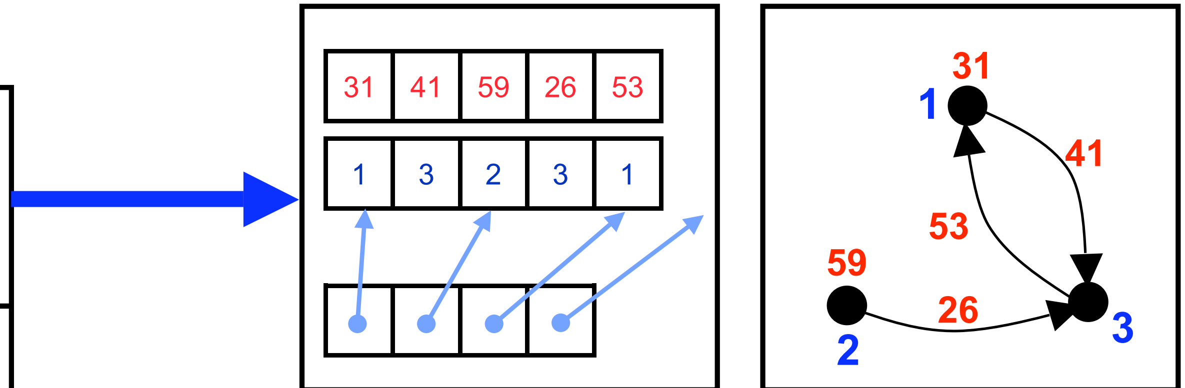


INTERACTIVE supercomputing

UCSB

Distributed sparse arrays





Each processor stores:

- # of local nonzeros (# local edges)
- range of local rows (local vertices)
- nonzeros in a compressed row data structure (local edges)

Sparse matrix operations

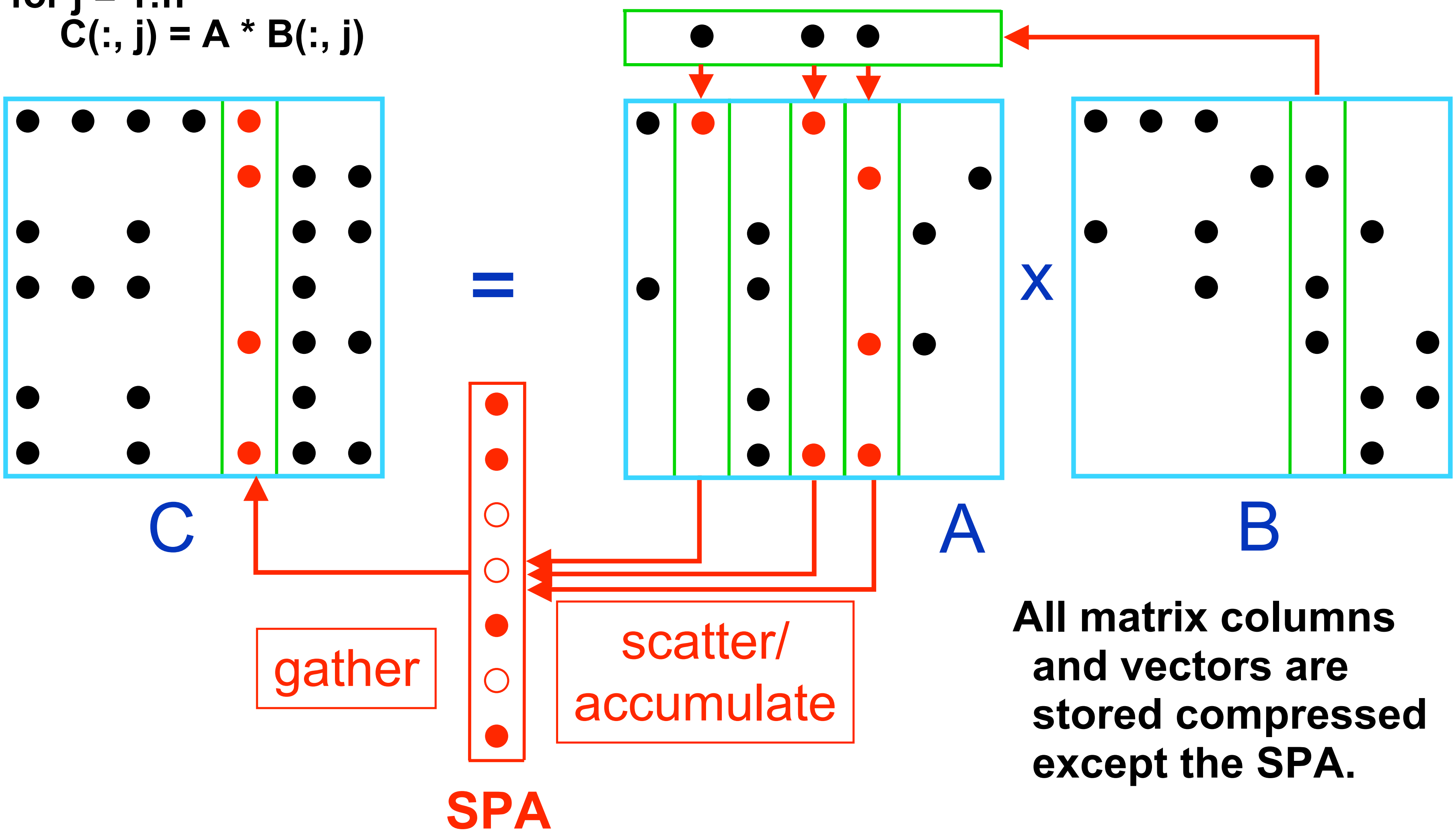
- dsparse layout, same semantics as ddense
- Matrix arithmetic: +, max, sum, etc.
- matrix * matrix and matrix * vector
- Matrix indexing and concatenation

  A (1:3, [4 5 2]) = [B(:, J) C] ;
- Linear solvers: x = A \ b; using MUMPS/SuperLU (MPI)
- Eigensolvers: [V, D] = eigs(A); using PARPACK (MPI)

Sparse matrix multiplication

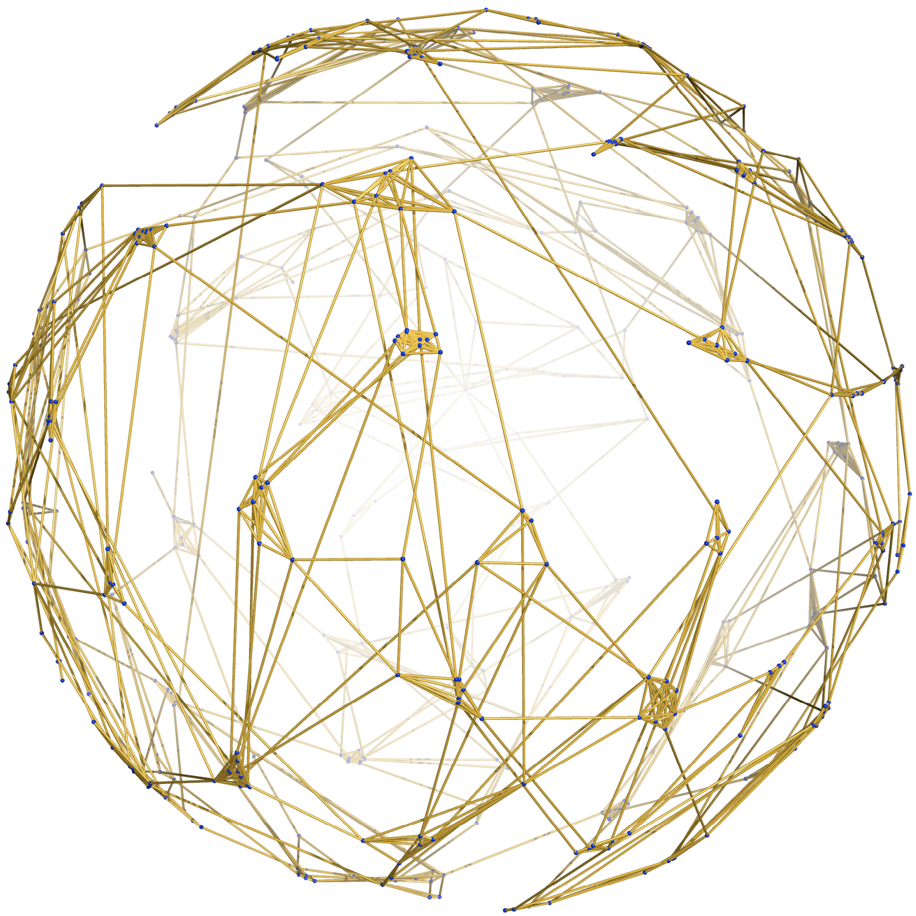
See A. Buluc (MS42, Fri 10am)

```
for j = 1:n
    C(:, j) = A * B(:, j)
```



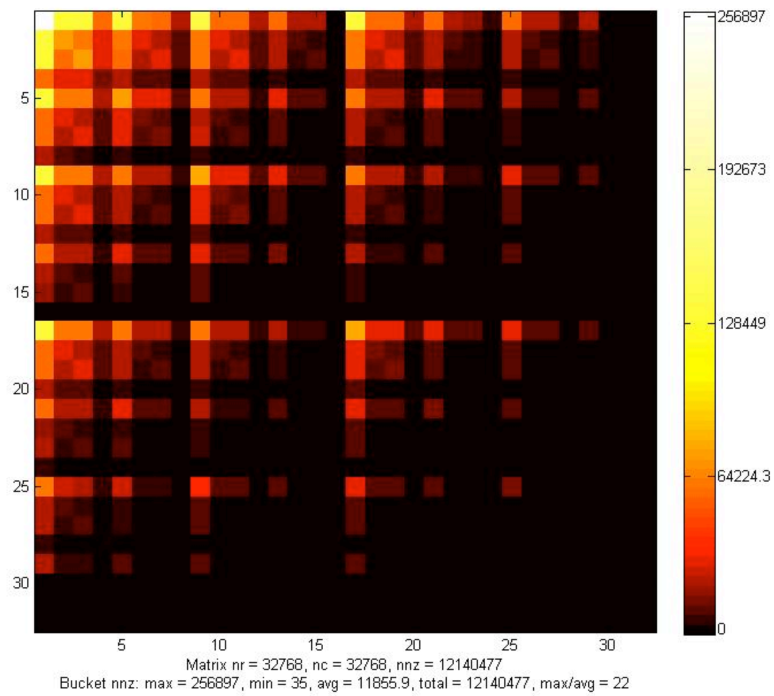
All matrix columns and vectors are stored compressed except the SPA.

Interactive data exploration



A graph plotted with relaxed Fiedler co-ordinates

A 2-D density spy plot

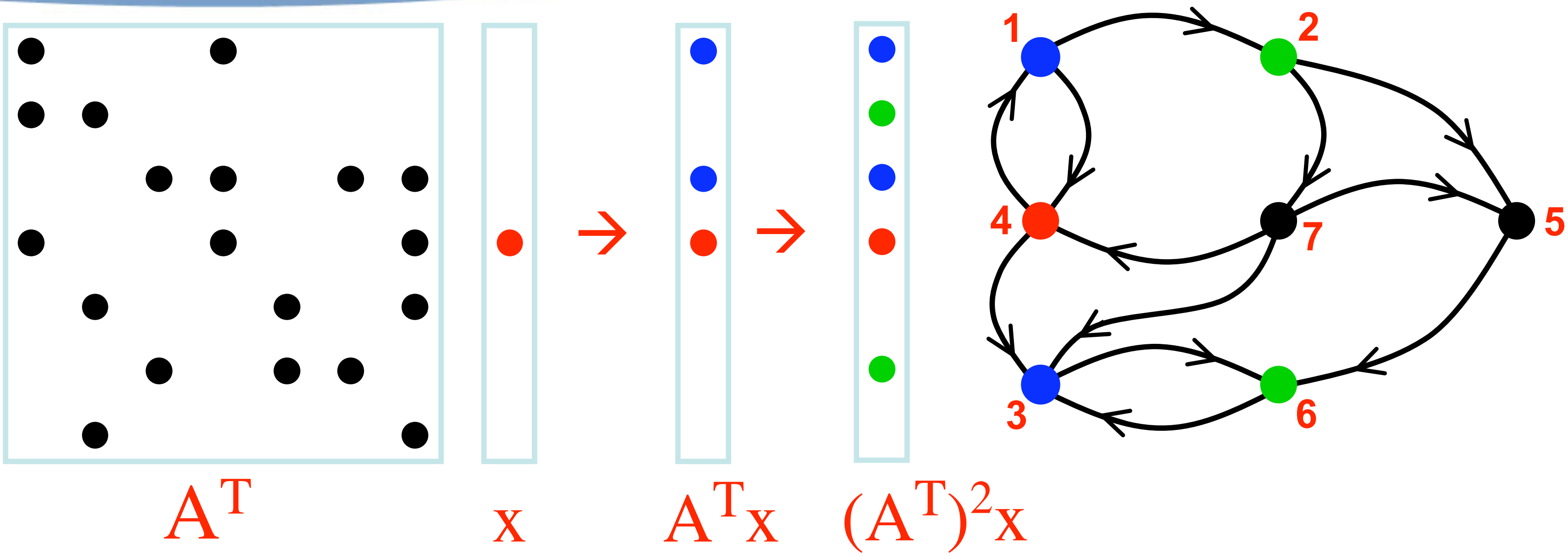


Density spy plot of an R-MAT power law graph

INTERACTIVE supercomputing

UCSB

Breadth-first search: sparse matvec



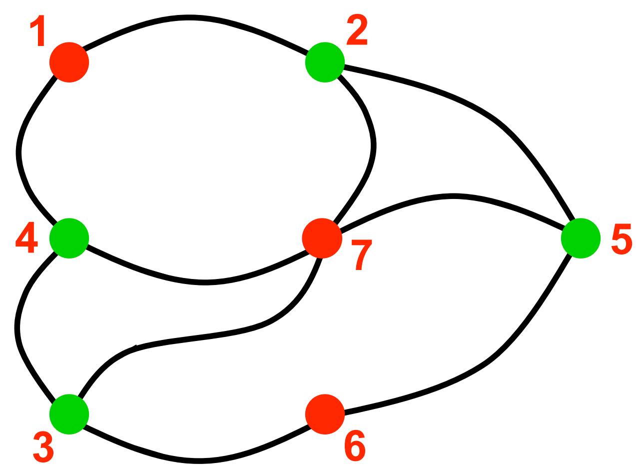
- Multiply by adjacency matrix → step to neighbor vertices
- Work-efficient implementation from sparse data structures

INTERACTIVE supercomputing

UCSB

Maximal independent set

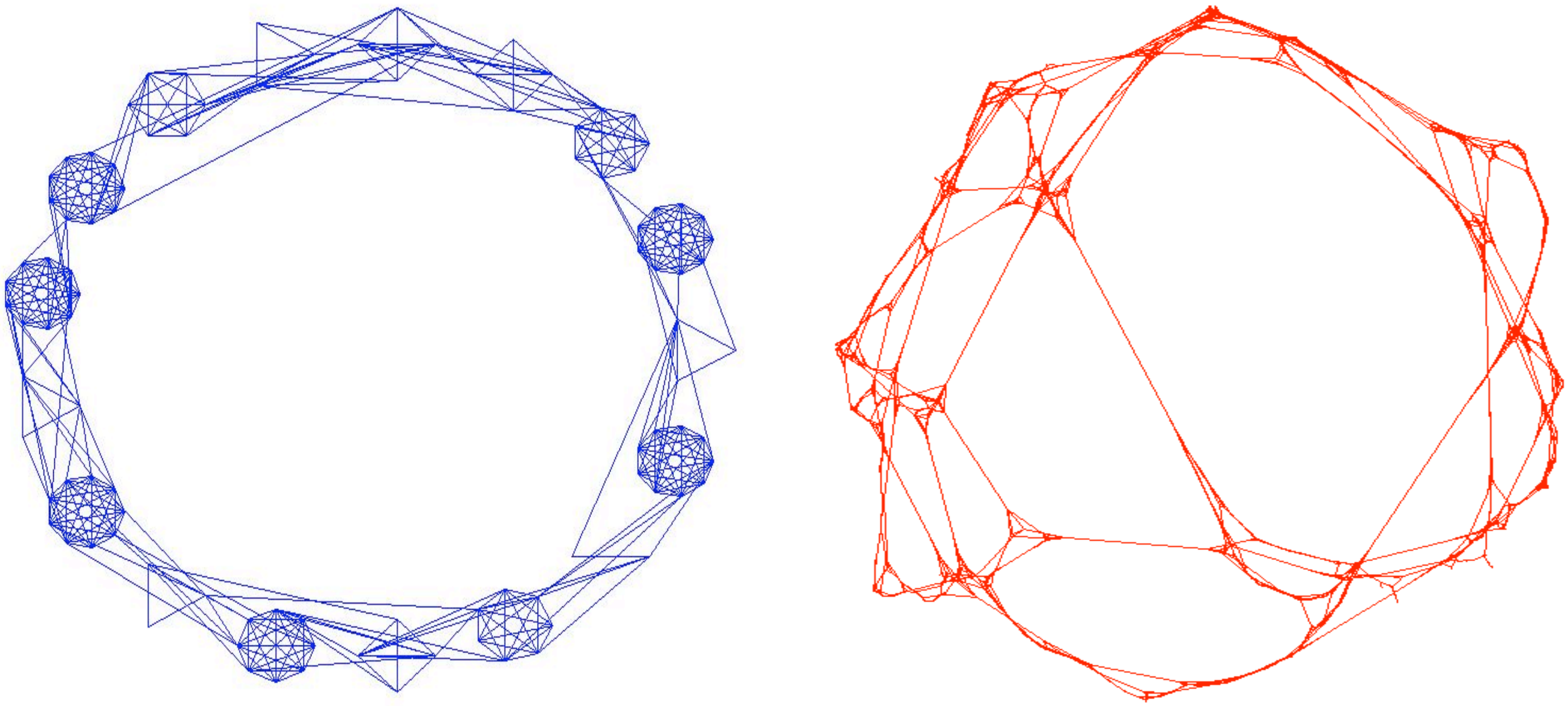
```
degree = sum(G, 2);
prob = 1 ./ (2 * deg);
select = rand (n, 1) < prob;

if ~isempty (select & (G * select);
  % keep higher degree vertices
end

IndepSet = [IndepSet select];

neighbor = neighbor | (G * select);
remain = neighbor == 0;
G = G(remain, remain);
```



Luby’s algorithm

A graph clustering benchmark

Fine-grained, irregular data access

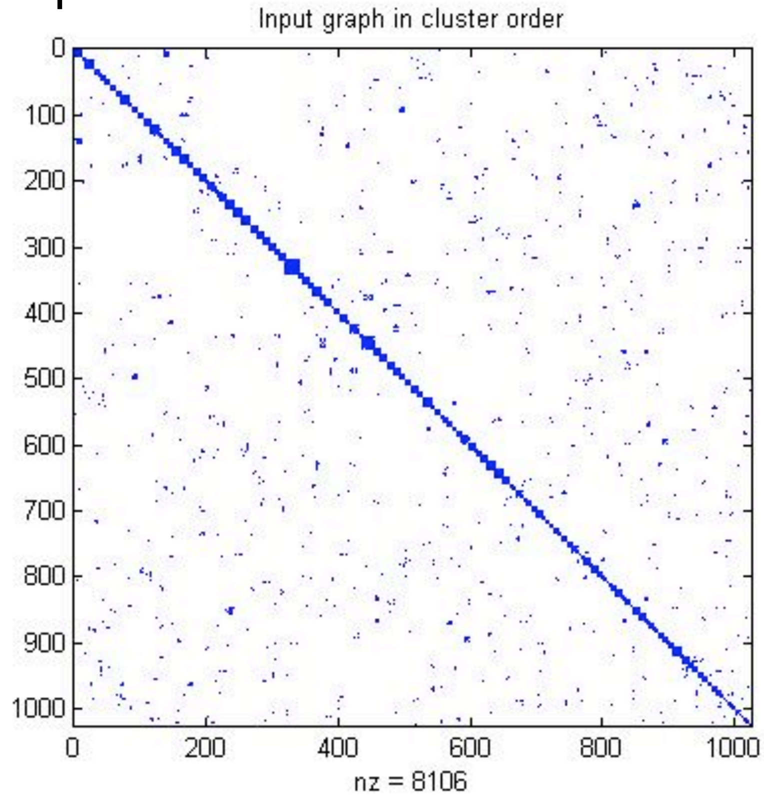


- Many tight clusters, loosely interconnected
- Vertices and edges permuted randomly

Clustering by BFS

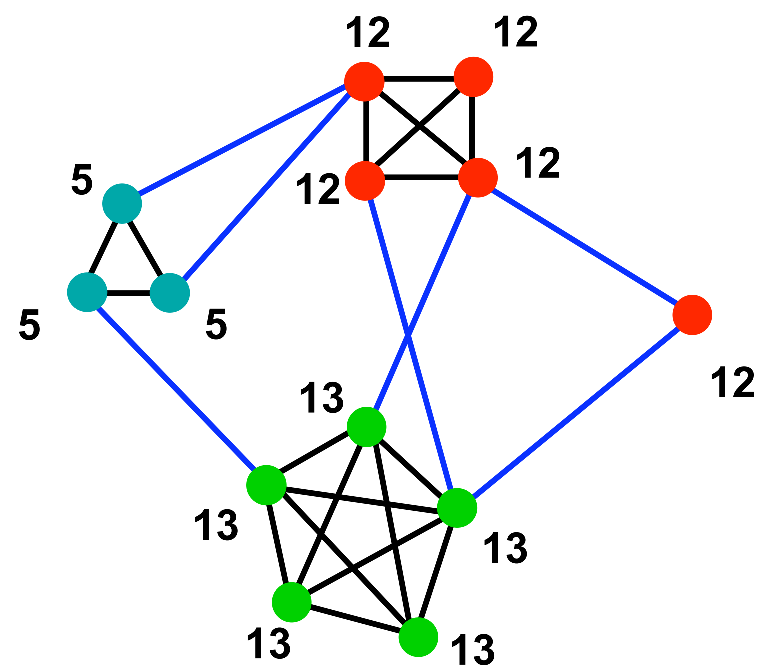
- Grow local clusters from many seeds in parallel
- Breadth-first search by sparse matrix * matrix
- Cluster vertices connected by many short paths

```
% Grow each seed to vertices
%    reached by at least k
%    paths of length 1 or 2

 C = sparse(seeds, 1:ns, 1, n, ns);
 C = A * C;
 C = C + A * C;
 C = C >= k;
```



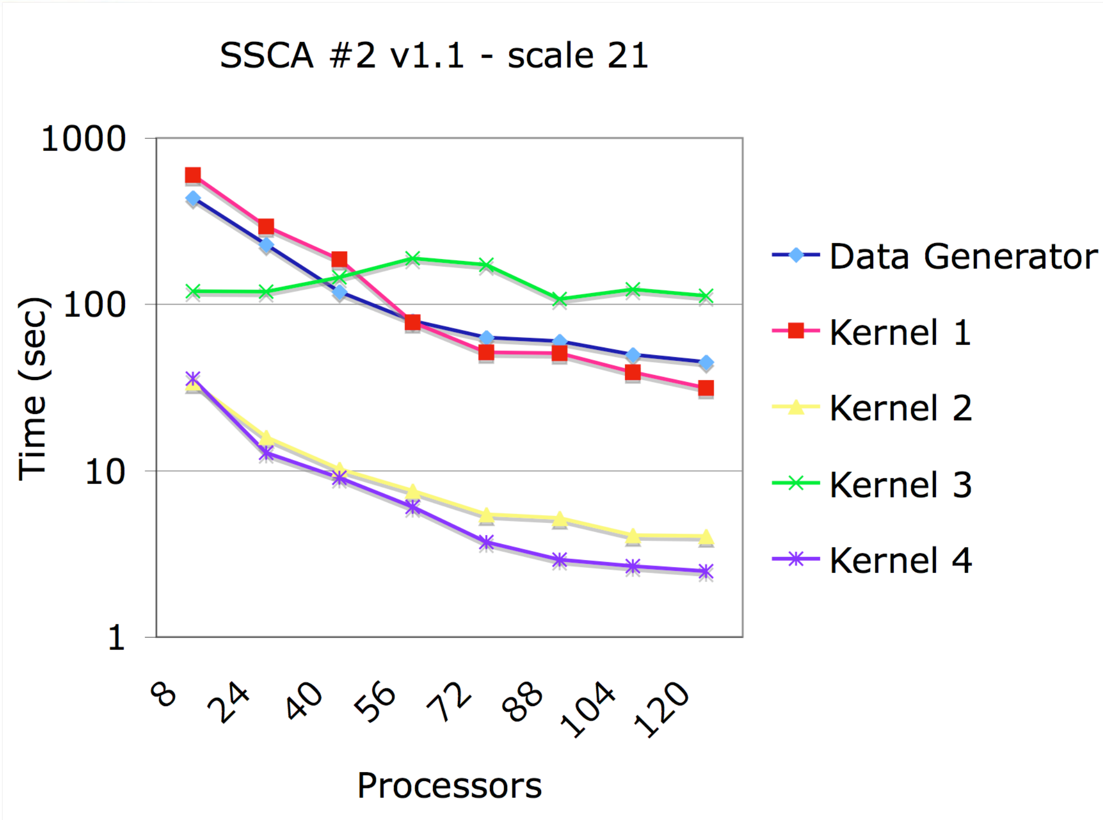
Clustering by peer pressure

```
[ignore, leader] = max(G);

S = sparse(leader,1:n,1,n,n) * G;

[ignore, leader] = max(S);
```



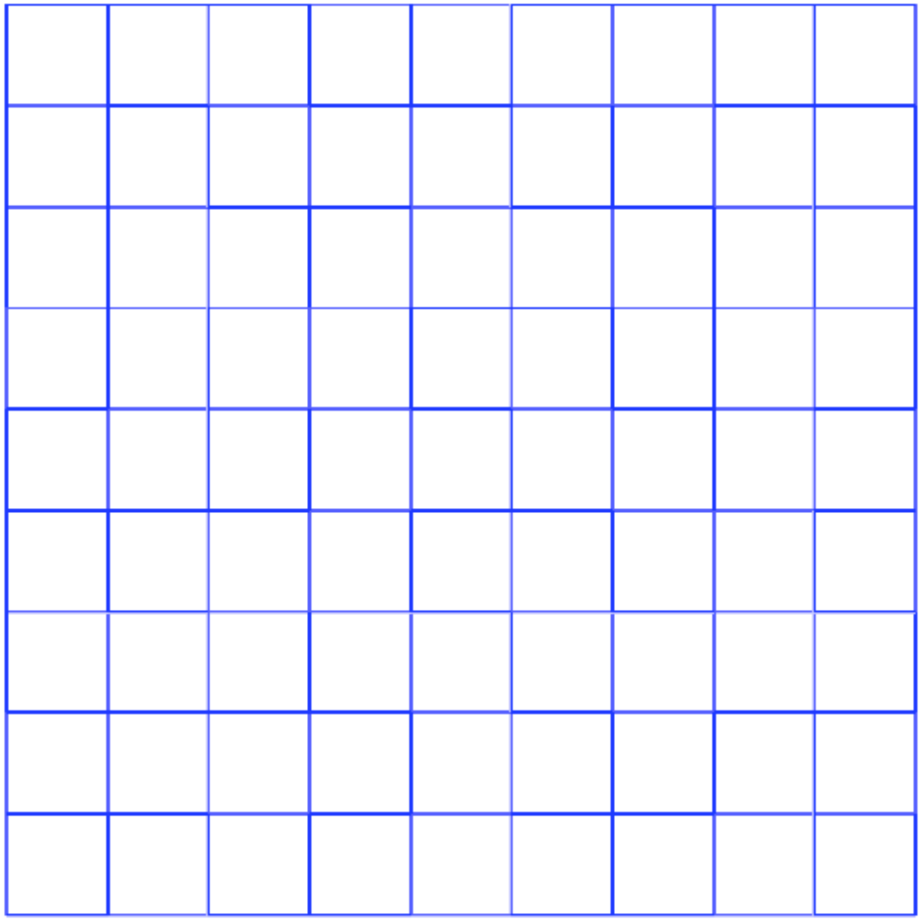
- Each vertex votes for its highest numbered neighbor as its leader
- Number of leaders is roughly the same as number of clusters
- Matrix multiplication gathers neighbor votes
- S(i,j) is the number of votes for i from j's neighbors

Scaling up



- Graph with 2 million nodes, 321 million directed edges, 89 million undirected edges, 32 thousand cliques
- Good scaling observed from 8 to 120 processors of an SGI Altix

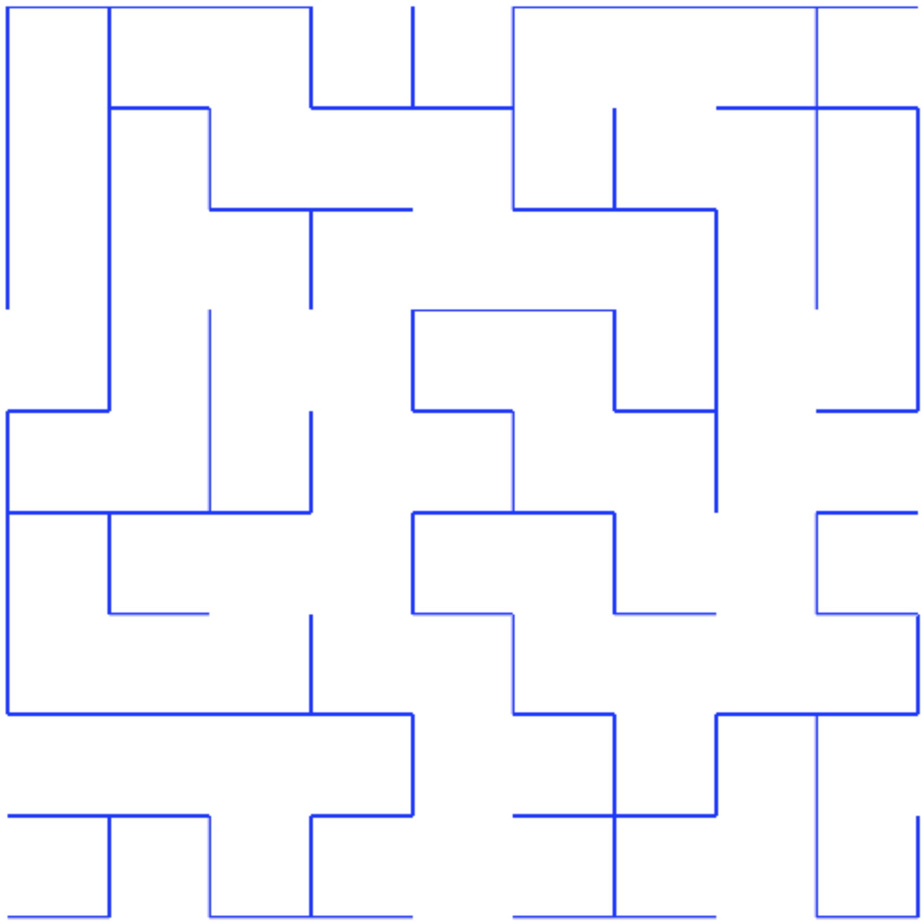
Graph Laplacian



Graph of Poisson's Equation on a 2D grid

G = grid5 (10);

Spanning trees

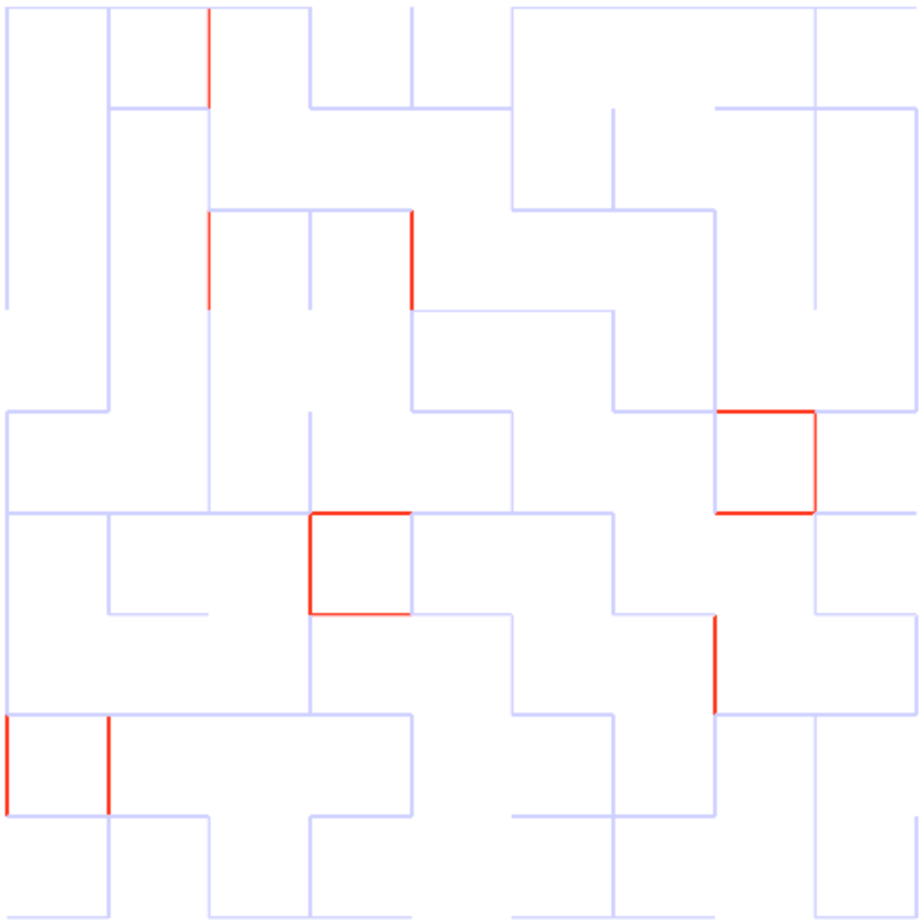


Maximum weight spanning tree

T = mst (G, 'max');

A combinatorial preconditioner

V. Aggarwal

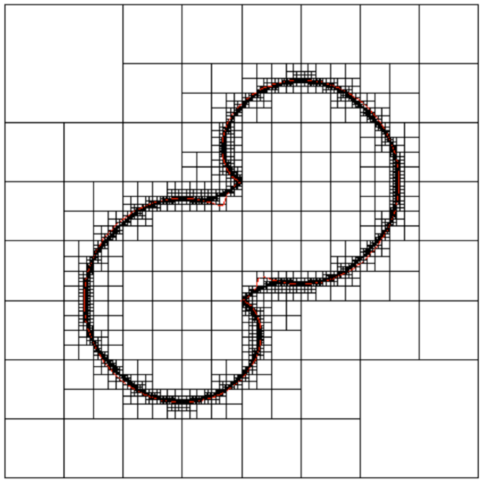


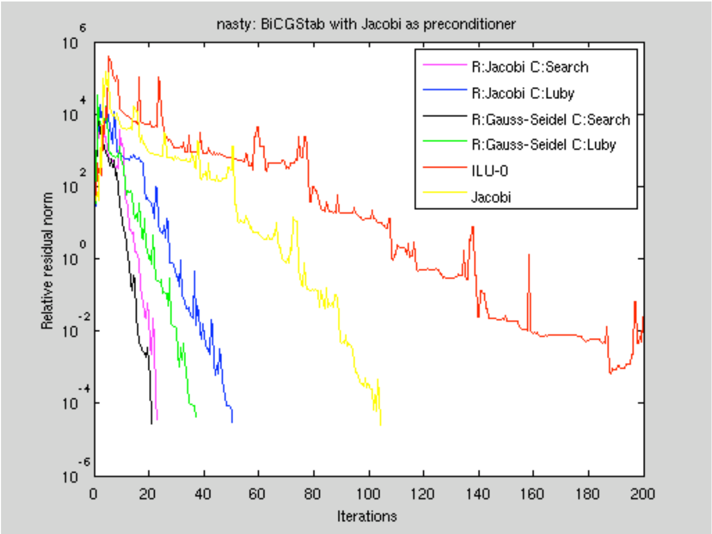
Augmented Vaidya's preconditioner

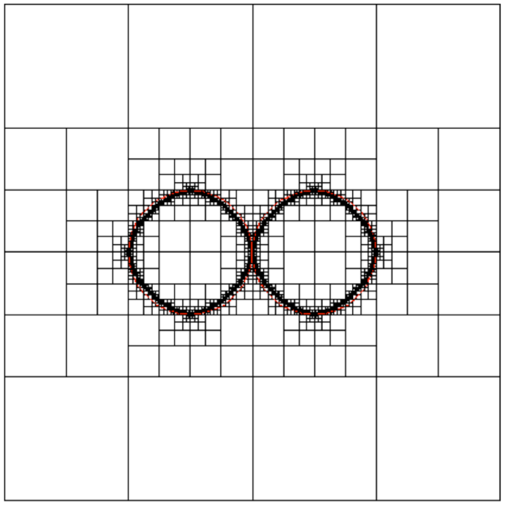
V = vaidya_support (G);

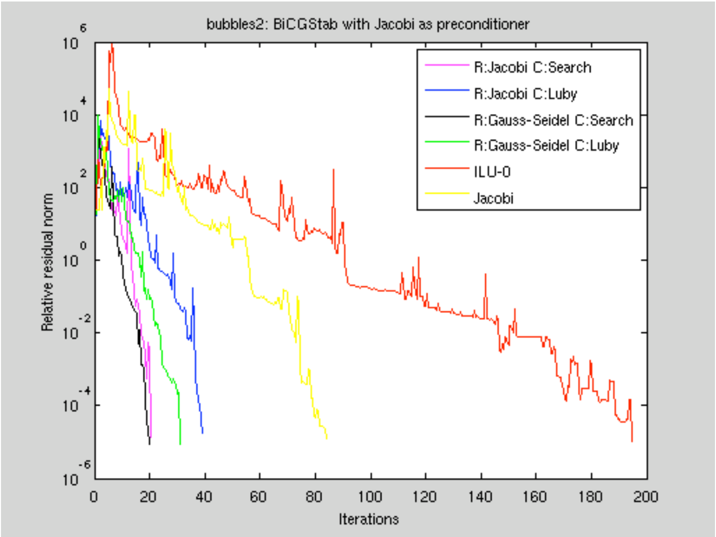
Quadtree meshes and AMG

V. Aggarwal and M. Roh









Wireless traffic modeling

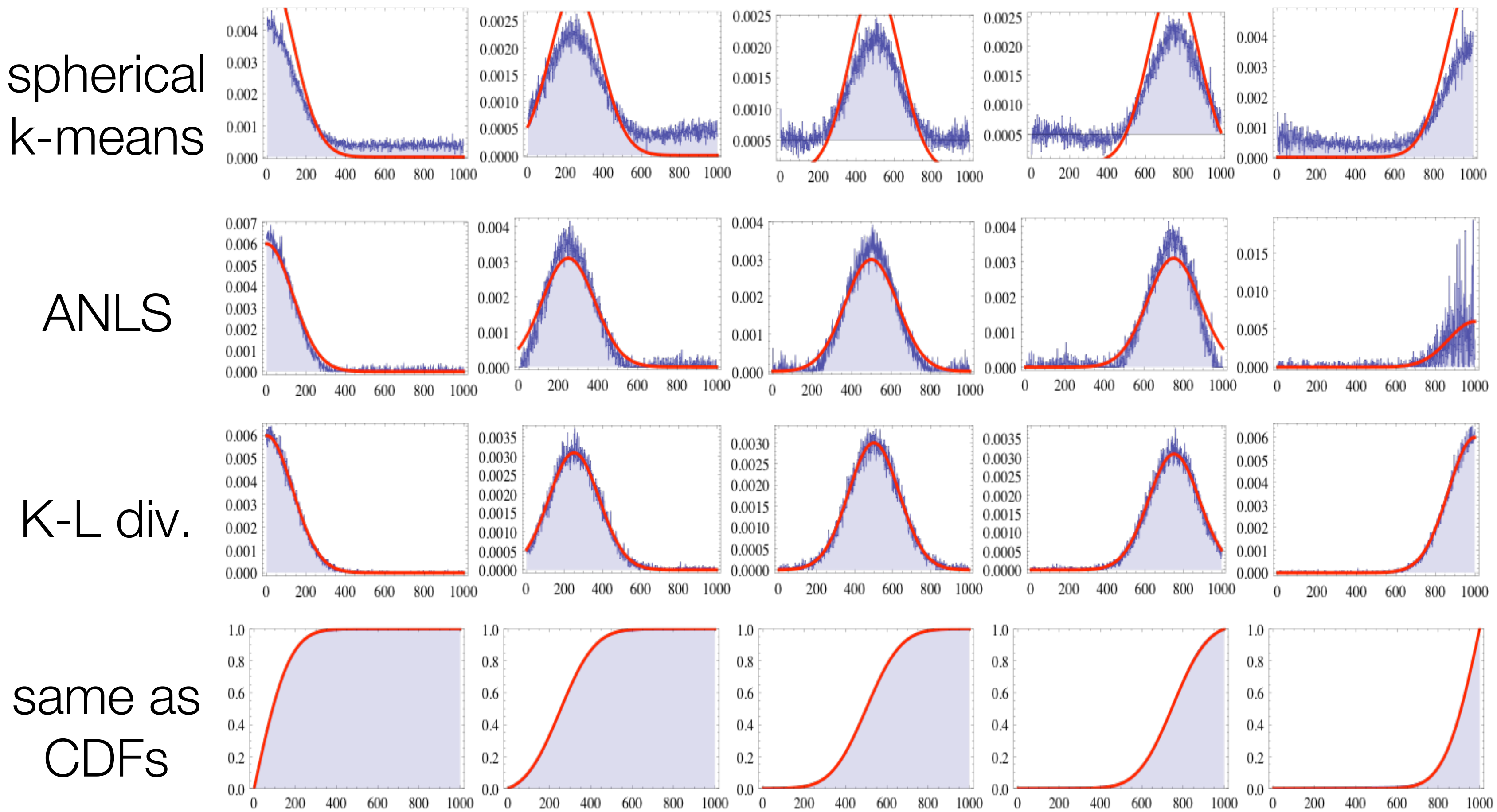
S. Karpinski

- Non-negative matrix factorizations (NNMF) for wireless traffic modeling

- NNMF algorithms combine linear algebra and optimization methods

- Basic and “improved” NMF factorization algorithms implemented:
  - euclidean (Lee & Seung 2000)
  - K-L divergence (Lee & Seung 2000)
  - semi-nonnegative (Ding et al. 2006)
  - left/right-orthogonal (Ding et al. 2006)
  - bi-orthogonal tri-factorization (Ding et al. 2006)
  - sparse euclidean (Hoyer et al. 2002)
  - sparse divergence (Liu et al. 2003)
  - non-smooth (Pascual-Montano et al. 2006)

A meta-algorithm

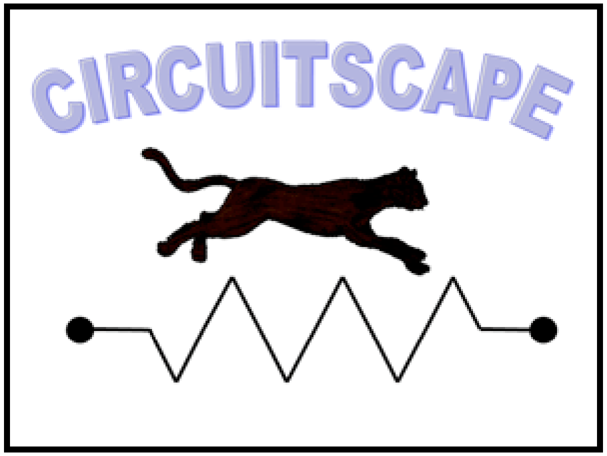


INTERACTIVE supercomputing

UCSB

Landscape Connectivity

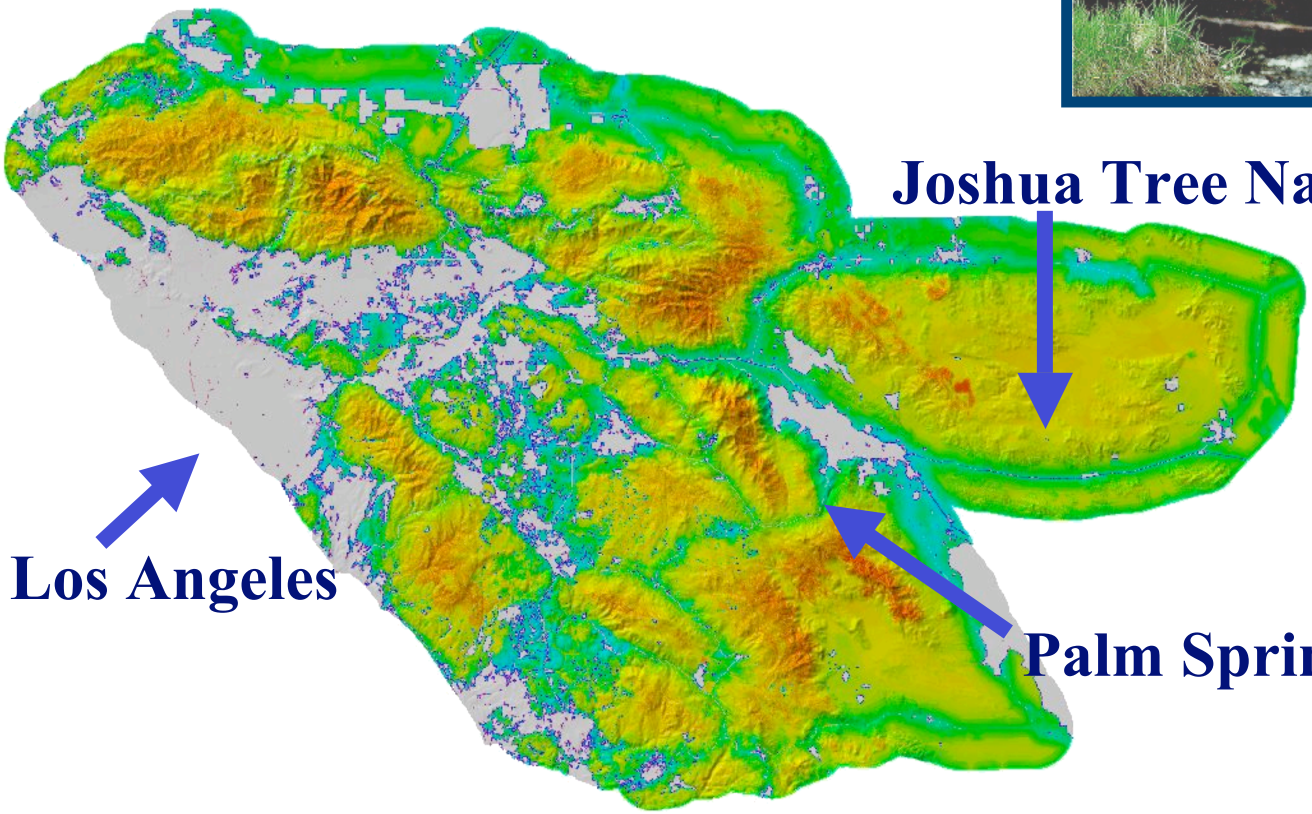
B. McRae



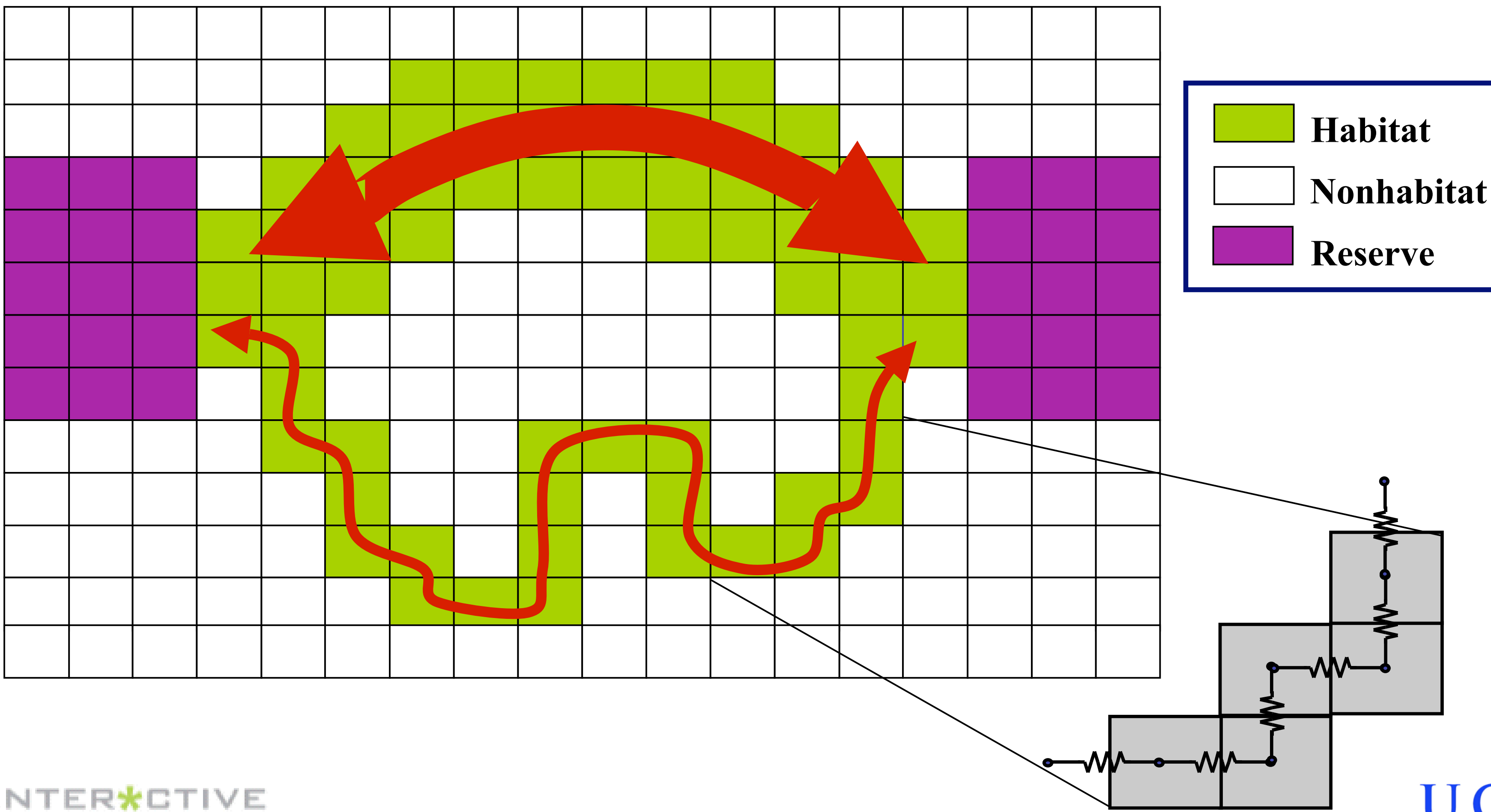
- Landscape connectivity governs the degree to which the landscape facilitates or impedes movement
- Need to model important processes like:
  - Gene flow (to avoid inbreeding)
  - Movement and mortality patterns
- Corridor identification, conservation planning

Pumas in southern California

Habitat quality model



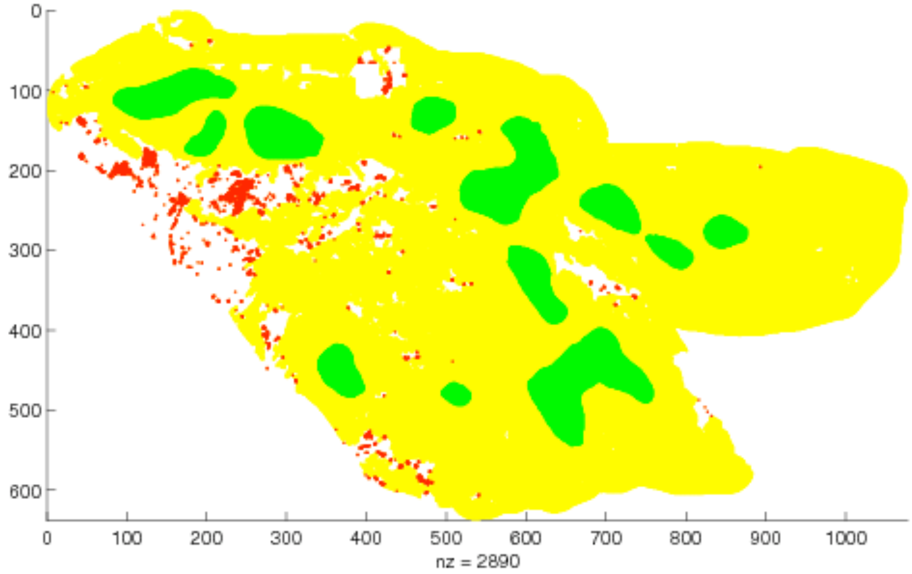
Model as a resistive network

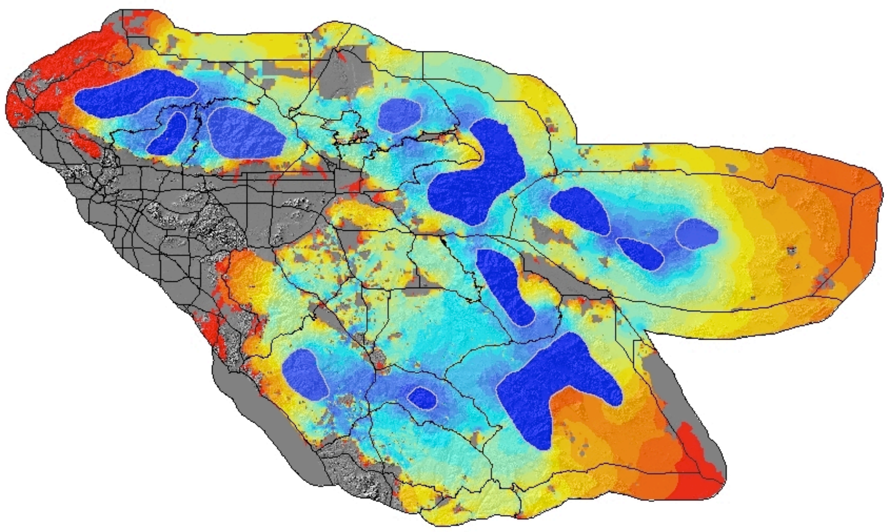


INTERACTIVE supercomputing

UCSB

Processing landscapes





Combinatorial methods

Graph construction

Graph contraction

Connected components

Numerical methods

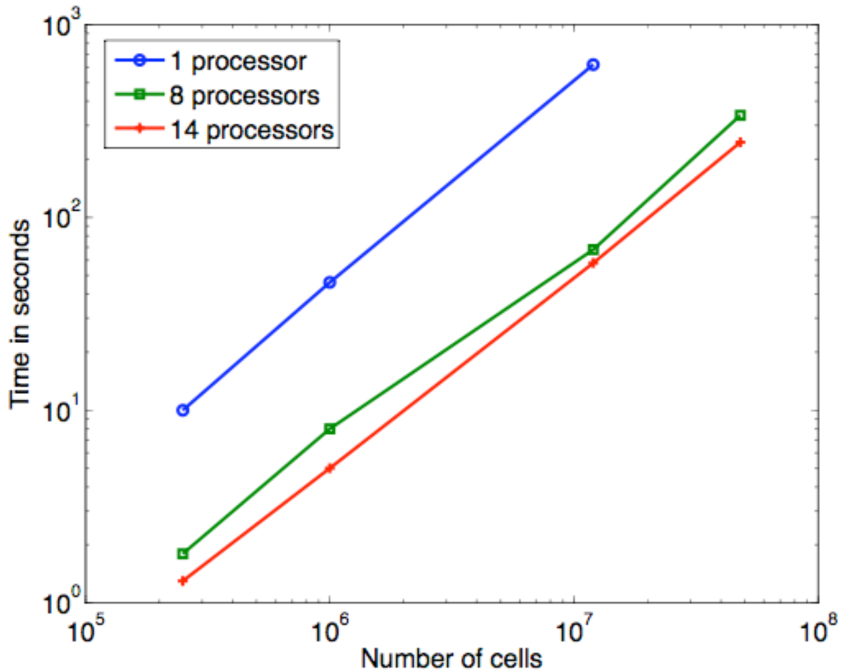
Linear systems

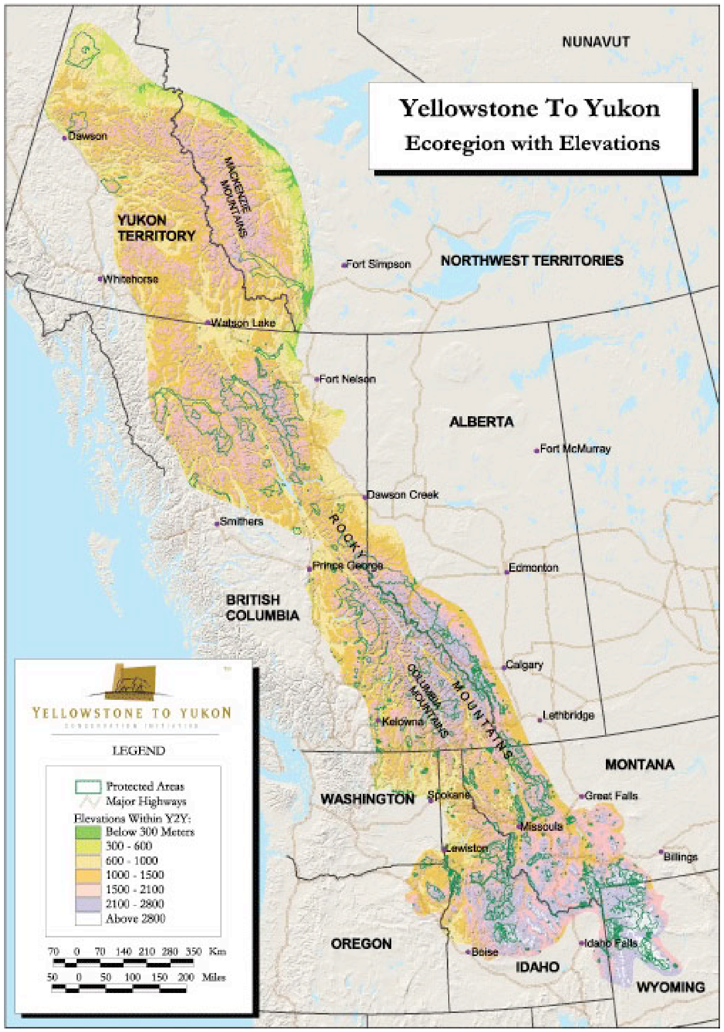
Combinatorial preconditioners

INTERACTIVE supercomputing

UCSB

Results





- Solution time reduced from 3 days to 5 minutes for typical problems
- Aiming for much larger problems: Yellowstone-to-Yukon (Y2Y)

INTERACTIVE supercomputing

UCSB

Multi-layered software tools

Computational ecology, CFD, data exploration **Applications**
CG, BiCGStab, etc. + combinatorial preconditioners (AMG, Vaidya) **Preconditioned Iterative Methods**
Graph querying & manipulation, connectivity, spanning trees, geometric partitioning, nested dissection, NNMF, . . . **Graph Analysis & PD Toolbox**
Arithmetic, matrix multiplication, indexing, solvers (\, eigs) **Distributed Sparse Matrices**

Thank You

Thanks for coming