STAR-P: High Productivity Parallel Computing

Ron Choy* Alan Edelman* John R. Gilbert† Viral Shah† David Cheng*

June 9, 2004

1 STAR-P

STAR-P ‡ is an interactive parallel scientific computing environment. It aims to make parallel programming more accessible. STAR-P borrows ideas from MATLAB*P [3], but is a new development. Currently only a MATLAB interface for STAR-P is available, but it is not limited to being a parallel MATLAB. It combines all four parallel MATLAB approaches in one environment, as described in the parallel MATLAB survey [2]: embarrassingly parallel, message passing, backend support and compilation. It also integrates several parallel numerical libraries into one single easy-to-use piece of software.

The focus of STAR-P is to improve user productivity in parallel programming. We believe that STAR-P can dramatically reduce the difficulty of programming parallel computers by reducing the time needed for development and debugging.

To achieve productivity, it is important that the user interface is intuitive to the user. For example, a large class of scientific users are already familiar with the MATLAB language. So it is beneficial to use MATLAB as a parallel programming language. Additions to the language are minimal in keeping with the philosophy to avoid re-learning. Also, as a design goal, our system does not distinguish between *serial* data and *parallel* data.

```
C = op(A,B)
print(C)
```

C will be the same whether A and B are distributed or not. This will allow the same piece of code to run sequentially (when all the arguments are serial) or in parallel (when at least one of the arguments is distributed).

* {cly,edelman,drcheng}@csail.mit.edu
† {gilbert,viral}@cs.ucsb.edu
‡ Some of this text appears in Ron Choy's upcoming Ph.D. thesis

2 Functionality

Where possible, STAR-P leverages existing, established parallel numerical libraries to perform numerical computation. This idea is inherited from MATLAB*P. Several libraries already exist which provide a wide range of linear algebra and other routines, and it would be inefficient to reproduce them. Instead, STAR-P integrates them seamlessly for the user.

3 RT-STAP Benchmarks

The RT-STAP (Real-Time Space-Time Adaptive Processing) benchmark [1] is a benchmark for real-time signal processing systems developed by the MITRE Corporation. In the hard version of the benchmark which we run, the input to the MATLAB code is a data cube of 22 (channels) x 64 x 480 doubles. The code performs the following three steps:

1. Convert the input data to baseband.
2. Doppler processing.
3. Weight computation and application to find the range-Doppler matrix.

Upon execution, we noticed that step 1 was the most time consuming step. This is surprising, since the weight computation would be expected to have the highest FLOP count. It turns out that this is due to the MATLAB coding style used in the benchmark code. Since the point of the benchmark is to measure the running time of a typical application, we chose to proceed without modifying the code. The conversion step in the original MATLAB code is a *for* loop as follows:

```
for channum=1:NCHAN
  xx = CPI1_INITIAL(:,channum);
  CPI1(:,channum) = baseband_convert(xx, ...
                    SOME_ARGUMENTS);
end
```

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
01 FEB 2005	**N/A**	**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Star-P: High Productivity Parallel Computing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology; University of California at Santa Barbara**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
See also ADM00001742, HPEC-7 Volume 1, Proceedings of the Eighth Annual High Performance Embedded Computing (HPEC) Workshops, 28-30 September 2004 Volume 1., The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**32**	

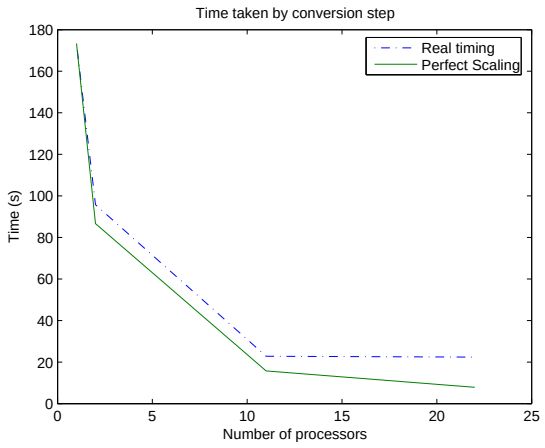


Figure 1: Scalability of RT_STAP

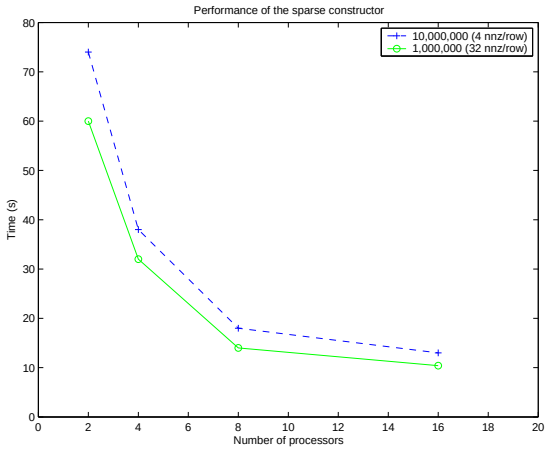


Figure 2: Scalability of *sparse* (SGI Altix 350)

It loops over the input channels and processes them in an embarrassingly parallel fashion. This makes it a natural candidate for Star-P's multi-Matlab mode. We converted the loop to run in Star-P by changing the loop into a function call and putting it in an mm-mode call:

```
P_CPI1_INITIAL = matlab2pp(CPI1_INITIAL,2);
CPI1 = mm('convert_loop', SOME_ARGUMENTS);
CPI1 = CPI1(:,:);
```

Note that the calls before and after the mm call are used to transfer data to the server and back. The time required by these calls is also included in our timings.

Figure 1 compares timing results for sequential Matlab and Star-P on 2, 11 and 22 processors. The solid line shows the timings that would be obtained if the code scales perfectly. The real timings follow the solid line quite closely except for the 22 processors case. Going from 11 processors to 22 processors provides no additional benefits. This is easy to explain in terms of granularity. As the input data cube only has 22 channels, with 22 processors, each processor has only 1 channel of work, as opposed to 2 channels in the 11 processors case. So there is very little to gain from using additional processors, and any benefit is offset by the additional time needed for communication.

4 Sparse matrix capabilities

Star-P provides basic sparse matrix capabilities [4] similar to those found in Matlab. Sparse matrix algorithms are often useful in signal processing and embedded computing. Sparse matrix operations often display poor spatial and temporal locality resulting in irregular memory access patterns.

A very basic and fundamental sparse matrix operation is the sparse matrix constructor in Matlab – *sparse*. It constructs a distributed sparse matrix from 3 vectors containing the row and column numbers and the corresponding non-zeros. The *sparse* constructor has fairly general applications in building and updating tables, histograms, and sparse data structures in general. It also accumulates and adds duplicate entries.

Figure 2 shows the performance of *sparse* on two matrices: one very large but sparse, with 10^7 rows and 4×10^7 non-zero entries; the other smaller and denser, with 10^6 rows and 32×10^6 non-zero entries.

References

[1] K. C. Cain, J. A. Torres, and R. T. Williams. RT_STAP: Real time space-time adaptive processing benchmark. Technical report, Feb 1997.

[2] R. Choy. Parallel Matlab survey. 2001. http://theory.lcs.mit.edu /∼cly/survey.html.

[3] P. Husbands and C. Isbell. MATLAB*P: A tool for interactive supercomputing. *SIAM PPSC*, 1999.

[4] V. Shah and J. R. Gilbert. Sparse Matrices in MATLAB*P: Design and Implementation. *Submitted to HiPC 2004*.

Abstract

Agenda

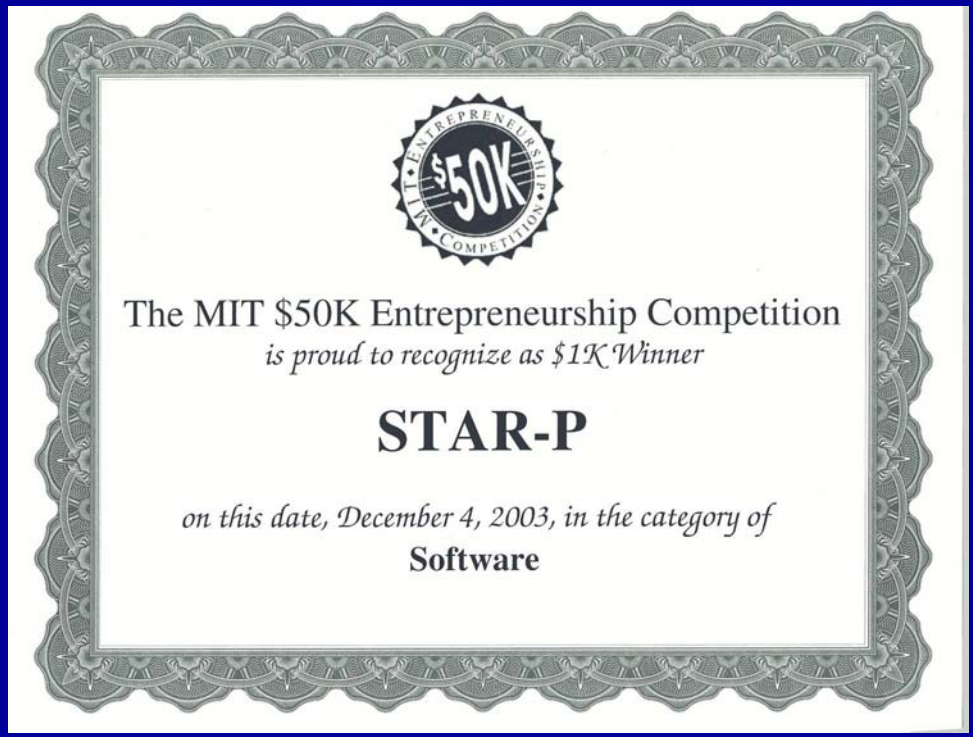
STAR-P: High Productivity Parallel Computing

David Cheng, Ron Choy, Alan Edelman
Massachusetts Institute of Technology

John R. Gilbert and Viral Shah
University of California at Santa Barbara

Graph Algorithms and Sparse Matrix Land

Birth of Interactive Supercomputing

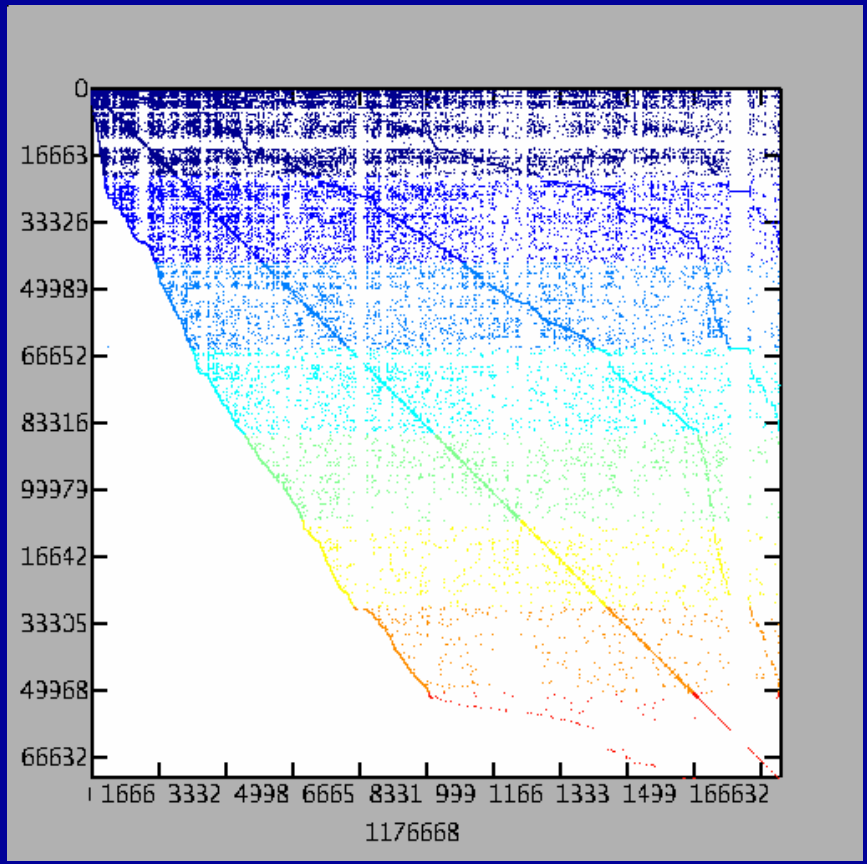


- Dream of taking academic software commercial

Star-P

- Interactive Parallel Computing Environment
- Parallel Client/Server Architecture
- Main goal: parallel computing easier on the human user
- Academic Front End: MATLAB
- Four parallel approaches interacting:
  - Embarrassingly Parallel
  - Message Passing
  - Backend Support (insert *p)
  - Compiling
- Integrates several packages into one easy to use software

Page Rank Matrix



- Web crawl of 170,000 pages from mit.edu
- Matlab*P spy plot of the matrix of the graph

Clock

- c=mm(‘clock’);
- std(c);

- Simple example shows two modes interacting

Pieces of Pi

```
>> quad('4./(1+x.^2)', 0, 1);
ans = 3.14159270703219

>> a = (0:3*p) / 4
a = ddense object: 1-by-4

>> a(:)
ans =
                    0
   0.25000000000000
   0.50000000000000
   0.75000000000000

>> b = a+.25;

>> c = mm('quad','4./(1+x.^2)', a, b);   % Should be “feval”!
c = ddense object: 1-by-4

>> sum(c(:))
ans = 3.14159265358979
```

FFT2 in four lines

```
>>  A = randn(4096, 4096*p)
A = ddense object: 4096-by-4096
>> tic;

>> B = mm('fft', A);
>> C = B.';
>> D = mm('fft', C);
>> F = D.';

>> toc
elapsed_time = 73.50

>>a = A(:,:);
>>  tic; g = fft2(a); toc
elapsed_time = 202.95
```

… we have FFTW installed as well!

Matlab sparse matrix design principles

- All operations should give the same results for sparse and full matrices (almost all)
- Sparse matrices are never created automatically, but once created they propagate
- Performance is important -- but usability, simplicity, completeness, and robustness are more important
- Storage for a sparse matrix should be O(nonzeros)
- Time for a sparse operation should be O(flops) (as nearly as possible)

Matlab sparse matrix design principles

- All operations should give the same results for sparse and full matrices (almost all)
- Sparse matrices are never created automatically, but once created they propagate
- Performance is important -- but usability, simplicity, completeness, and robustness are more important
- Storage for a sparse matrix should be O(nonzeros)
- Time for a sparse operation should be O(flops) (as nearly as possible)

Matlab*P dsparse matrices: same principles, but some different tradeoffs

Sparse matrix operations

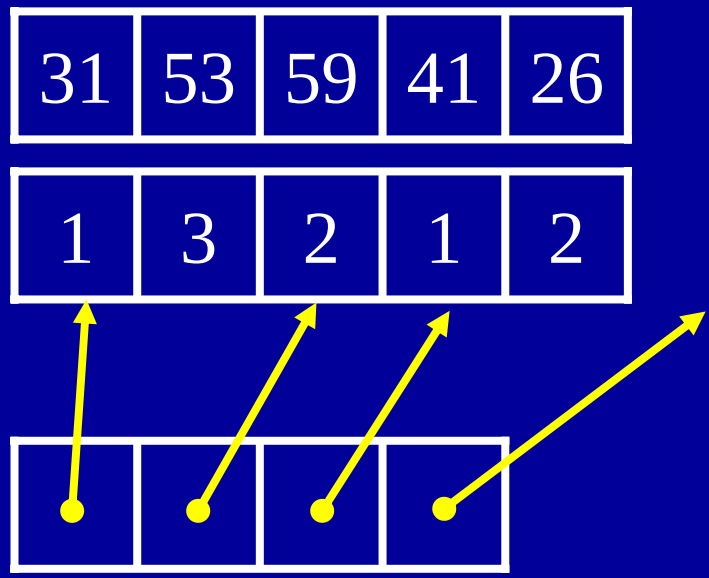
- dsparse layout, same semantics as ddense
- For now, only row distribution
- Matrix operators: +, -, max, etc.
- Matrix indexing and concatenation

```
A (1:3, [4 5 2]) = [ B(:, 7) C ] ;
```

- A \ b by direct methods
- Conjugate gradients

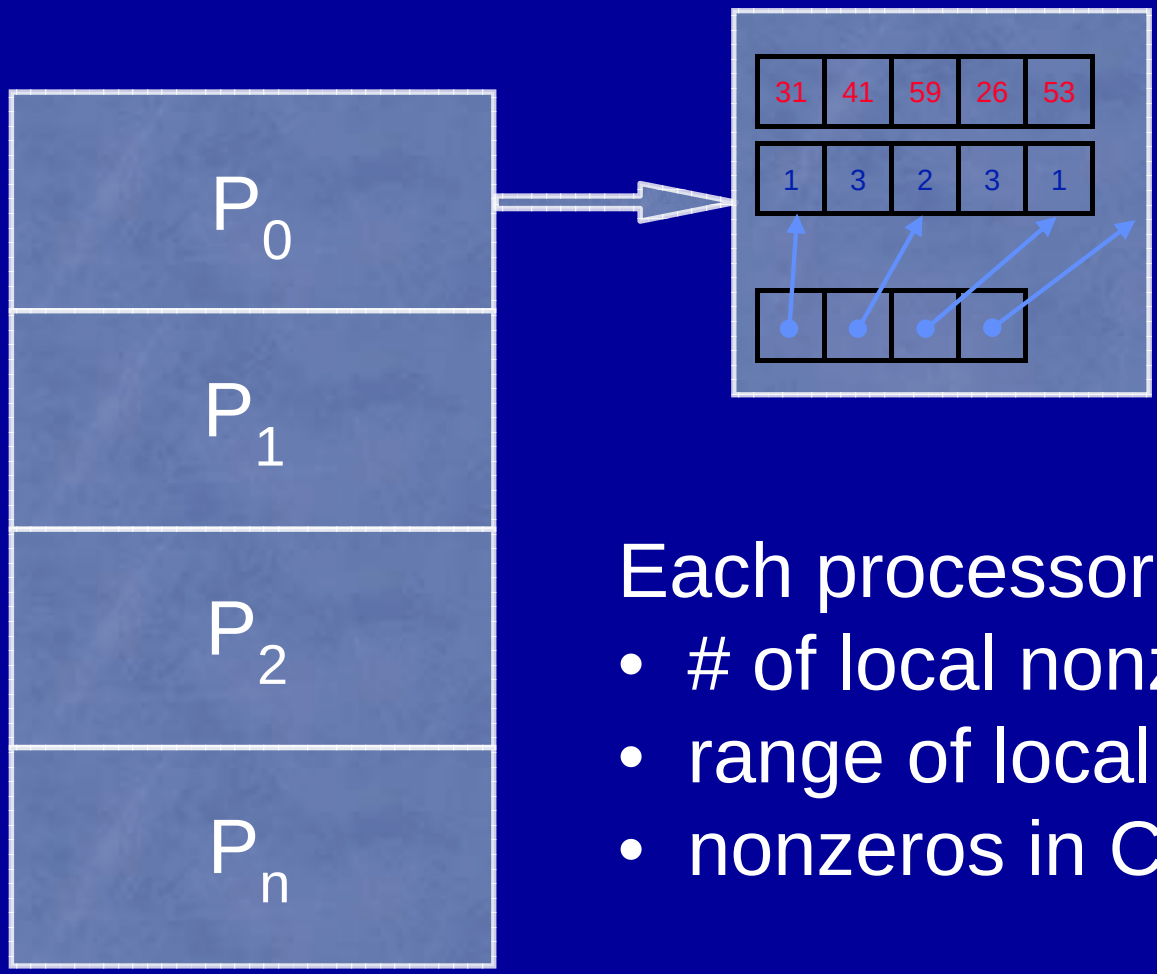
Sparse data structure

31	0	53
0	59	0
41	26	0



- Full:
  - 2-dimensional array of real or complex numbers
  - (nrows*ncols) memory
- Sparse:
  - compressed row storage
  - about (1.5*nzs + .5*nrows) memory

Distributed sparse data structure



Each processor stores:

- # of local nonzeros
- range of local rows
- nonzeros in CSR form

Sparse matrix times dense vector

- y = A * x
- The first call to matvec caches a communication schedule for matrix A. Later calls to multiply any vector by A use the cached schedule.
- Communication and computation overlap.
- Can use a tuned sequential matvec kernel on each processor.

Sparse linear systems

- x = A \ b
- Matrix division uses MPI-based direct solvers:
  - SuperLU_dist: nonsymmetric static pivoting
  - MUMPS: nonsymmetric multifrontal
  - PSPASES: Cholesky

```
ppsetoption('SparseDirectSolver','SUPERLU')
```

- Iterative solvers implemented in Matlab*P
- Some preconditioners; ongoing work

Application: Fluid dynamics

- Modeling density-driven instabilities in miscible fluids (Goyal, Meiburg)
- Groundwater modeling, oil recovery, etc.
- Mixed finite difference & spectral method
- Large sparse generalized eigenvalue problem

```
function lambda = peigs (A, B,
  sigma, iter, tol)

  [m n] = size (A);
  C = A - sigma * B;
  y = rand (m, 1);

  for k = 1:iter
    q = y ./ norm (y);
    v = B * q;
    y = C \ v;
    theta = dot (q, y);
    res = norm (y - theta*q);
    if res <= tol
      break;
    end;
  end;

  lambda = 1 / theta;
```

Combinatorial algorithms in Matlab*P

- Sparse matrices are a good start on primitives for combinatorial scientific computing.
  - Random-access indexing: `A(i,j)`
  - Neighbor sequencing: `find (A(i,:))`
  - Sparse table construction: `sparse (I, J, V)`
- What else do we need?

Sorting in Matlab*P

- `[V, perm] = sort (V)`
- Common primitive for many sparse matrix and array algorithms: sparse(), indexing, transpose
- Matlab*P uses a parallel sample sort

Sample sort

- (Perform a random permutation)
- Select p-1 “splitters” to form p buckets
- Route each element to the correct bucket
- Sort each bucket locally
- “Starch” the result to match the distribution of the input vector

Sample sort example

Initial data (after randomizing)

Choose splitters (2 and 6)

Sort local data

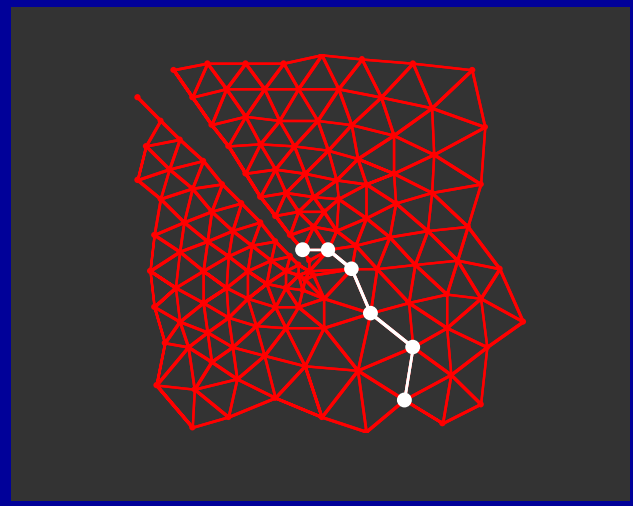
Starch

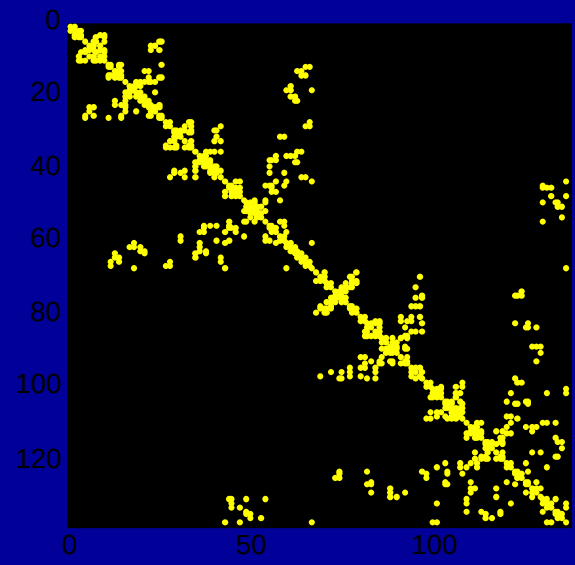
How sparse( ) works

- **`A = sparse (I, J, V)`**
- Input: ddense vectors I, J, V (optionally, also dimensions and distribution info)
- Sort triples (i, j, v) by (i, j)
- Starch the vectors for desired row distribution
- Locally convert to compressed row indices
- Sum values with duplicate indices

Graph / mesh partitioning

- Reduce communication in matvec and other parallel computations
- Reordering for sparse GE
- PARMETIS
- Parts of G/Teng Matlab meshpart toolbox





Geometric mesh partitioning

- Algorithm of Miller, Teng, Thurston, Vavasis
- Partitions irregular finite element meshes into equal-size pieces with few connecting edges
- Guaranteed quality partitions for well-shaped meshes, often very good results in practice
- Existing implementation in sequential Matlab
- Code runs in Matlab*P with very minor changes

Outline of algorithm

1. Project points stereographically from R^d to R^{d+1}
2. Find “centerpoint” (generalized median)
3. Conformal map: Rotate and dilate
4. Find great circle
5. Unmap and project down
6. Convert circle to separator

Geometric mesh partitioning

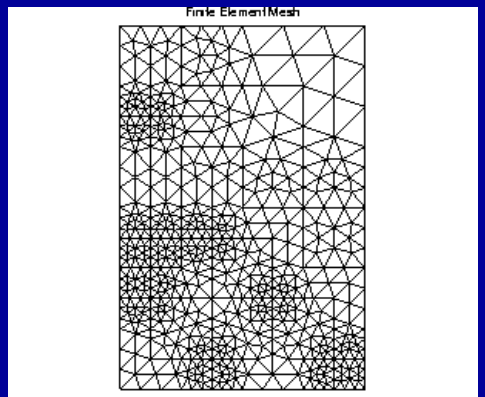


Figure 1: The input mesh.

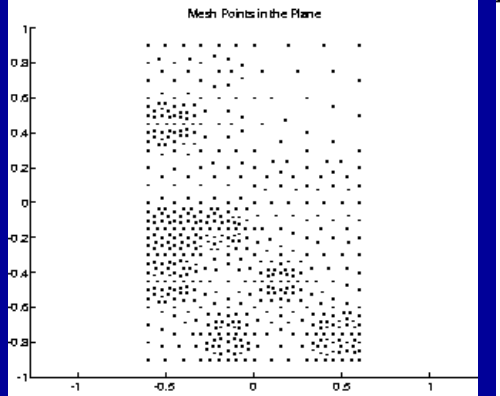


Figure 2: The mesh points.

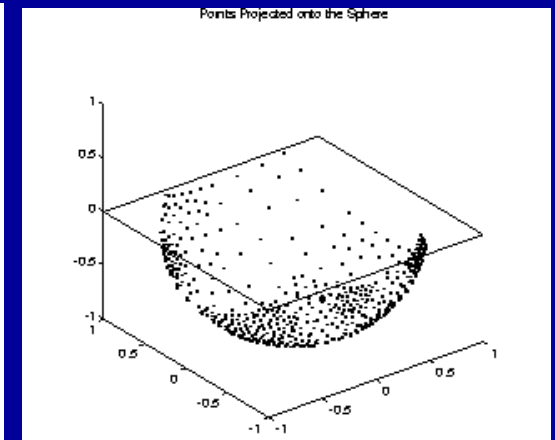
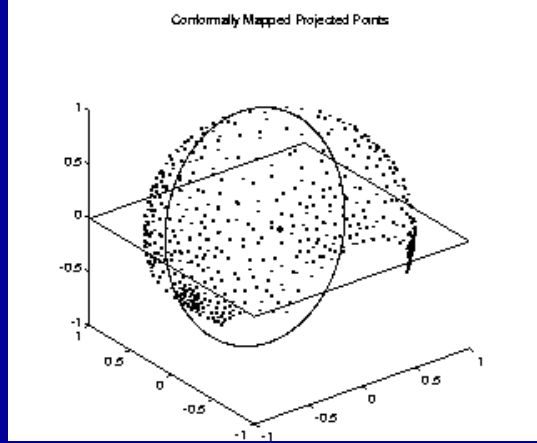


Figure 3: Projected mesh points. The large dot is the centerpoint.



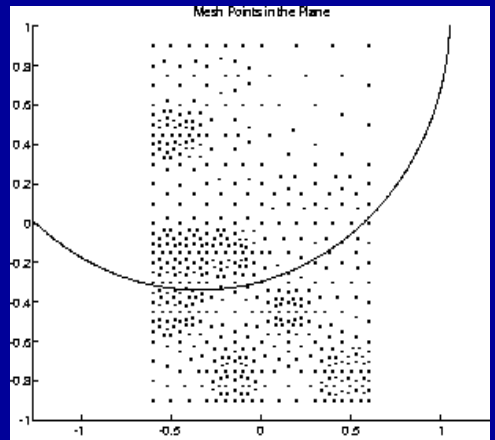
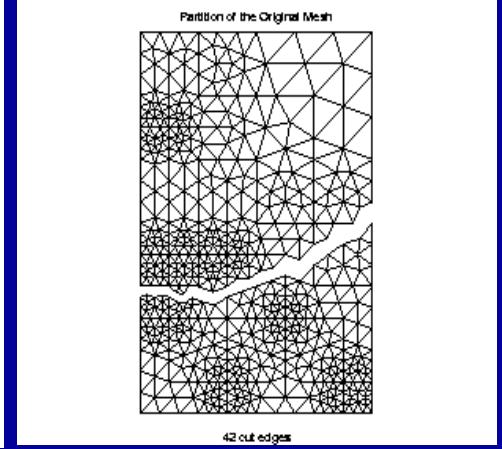


Figure 5: The separating circle projected back to the plane.



Matching and depth-first search in Matlab

- dmperm: Dulmage-Mendelsohn decomposition

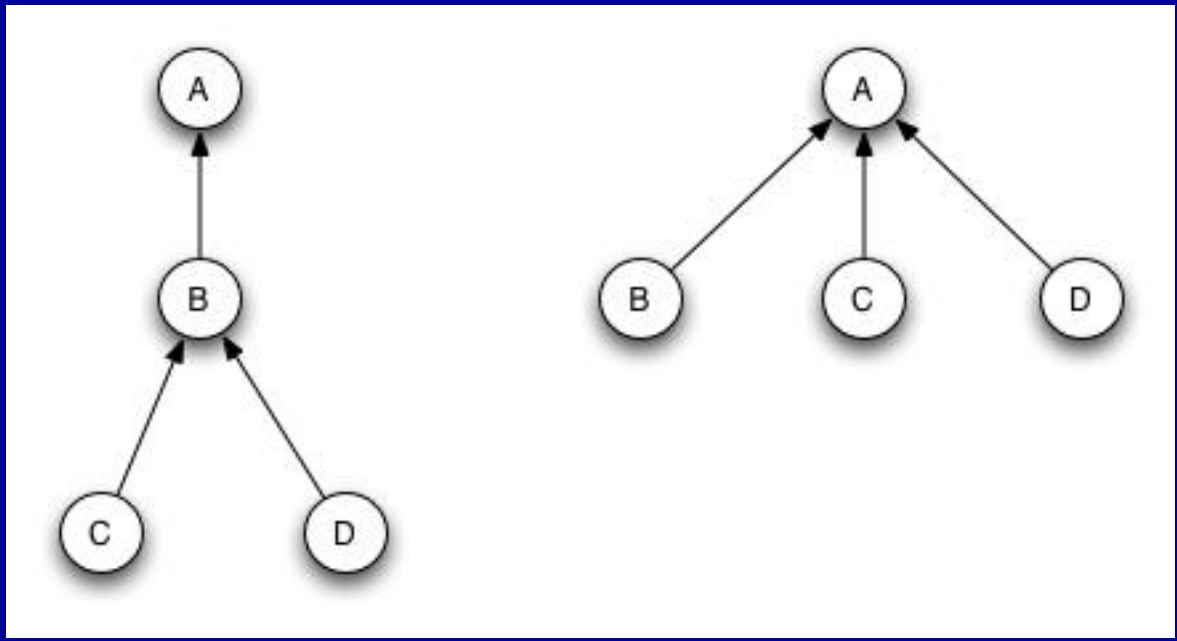
- Square, full rank A:
  - [p, q, r] = dmperm(A);
  - A(p,q) is block upper triangular with nonzero diagonal
  - also, strongly connected components of a directed graph
  - also, connected components of an undirected graph

- Arbitrary A:
  - [p, q, r, s] = dmperm(A);
  - maximum-size matching in a bipartite graph
  - minimum-size vertex cover in a bipartite graph
  - decomposition into strong Hall blocks

Connected components

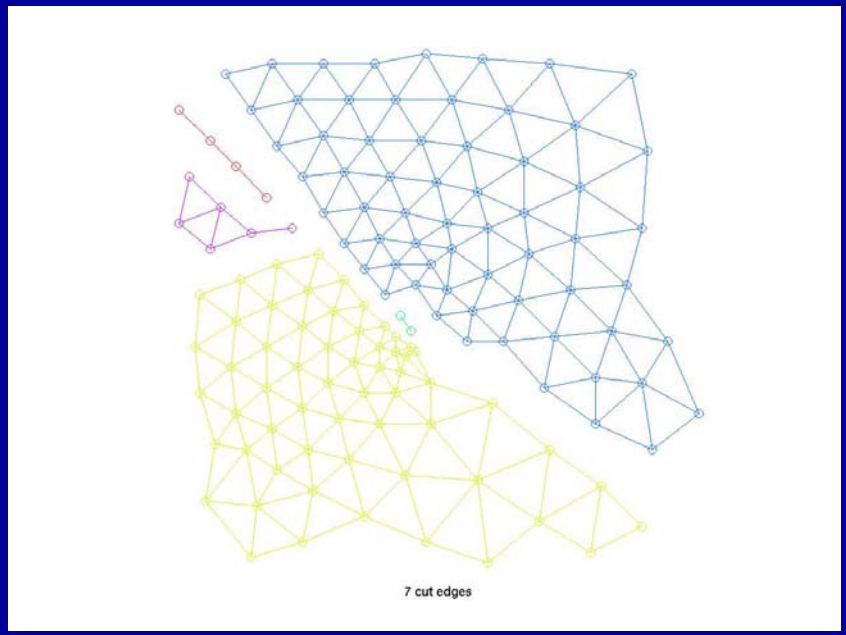
- Sequential Matlab uses depth-first search (`dmperm`), which doesn't parallelize well
- Shiloach-Vishkin algorithm:
  - repeat
    - Link every (super)vertex to a random neighbor
    - Shrink each tree to a supervertex by pointer jumping
  - until no further change
- Originally a processor-efficient PRAM algorithm
- Matlab*P code looks much like the PRAM code

Pointer jumping



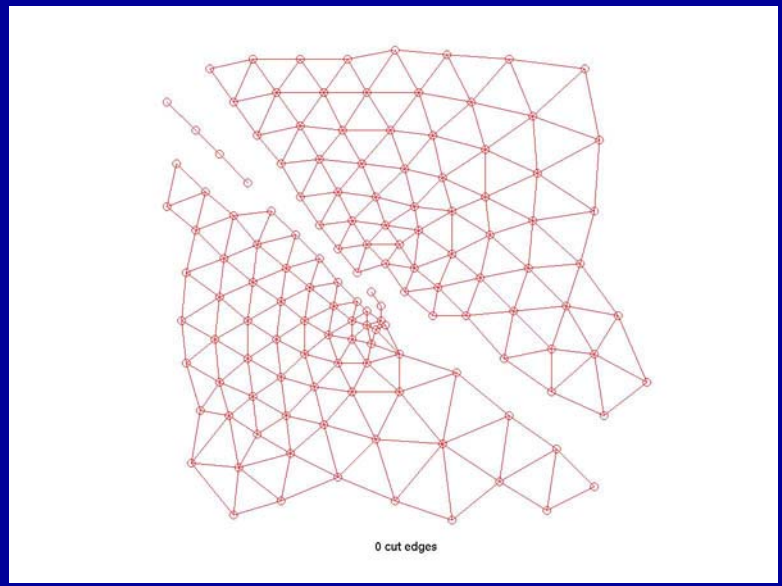

```
while ~all( C(myrows) == C(C(myrows)) )
  C(myrows) = C(C(myrows));
end
C(myrows) = min (C(myrows), C(C(myrows)));
```

Example of execution



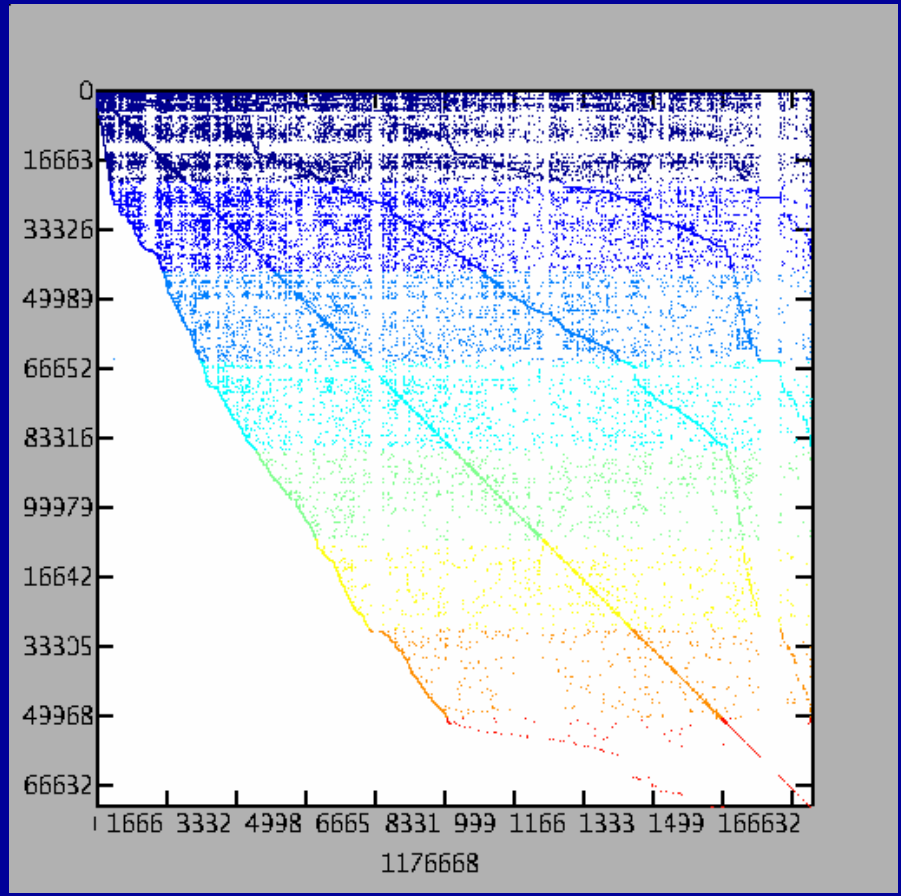
After first iteration

Final components



Page Rank

- Importance ranking of web pages
- Stationary distribution of a Markov chain
- Power method: matvec and vector arithmetic
- Matlab*P page ranking demo (from SC'03) on a web crawl of mit.edu (170,000 pages)



Remarks

- Easy-to-use interactive programming environment
- Interface to existing parallel packages
- Combinatorial methods toolbox being built on parallel sparse matrix infrastructure
  - Much to be done: spanning trees, searches, etc.
- A few issues for ongoing work
  - Dynamic resource management
  - Fault management
  - Programming in the large