Interactive Supercomputing

Delivering Interactive Parallel Computing Power to the Desktop

sgi

An Interactive Approach to Parallel Computing Algorithms with Star-P

Alan Edelman MIT (& ISC)

Contact:Alan Edelman

Interactive Supercomputing • 135 Beaver Street, FL 2 • Waltham, MA 02452

781.398-0010 • edelman@interactivesupercomputing.com • www.interactivesupercomputing.com

Parallel Computing Arts

Message Passing:

The King's Messenger

Batch Processing:

Coding, Modeling, & Debugging

Punch Cards (textile loom 1840)

Noble perfected arts: what's next for productivity?

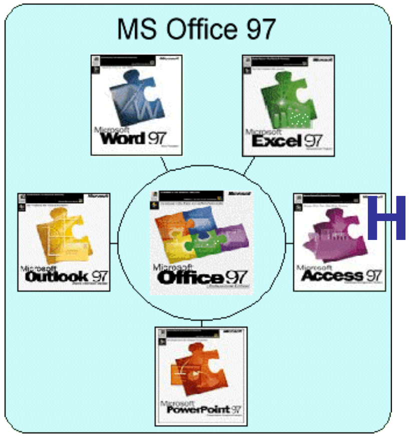
Interactive Supercomputing & sgi

Productivity

← Make this machine go faster?

Most important catalysts for productivity are

Interactivity & ease of use

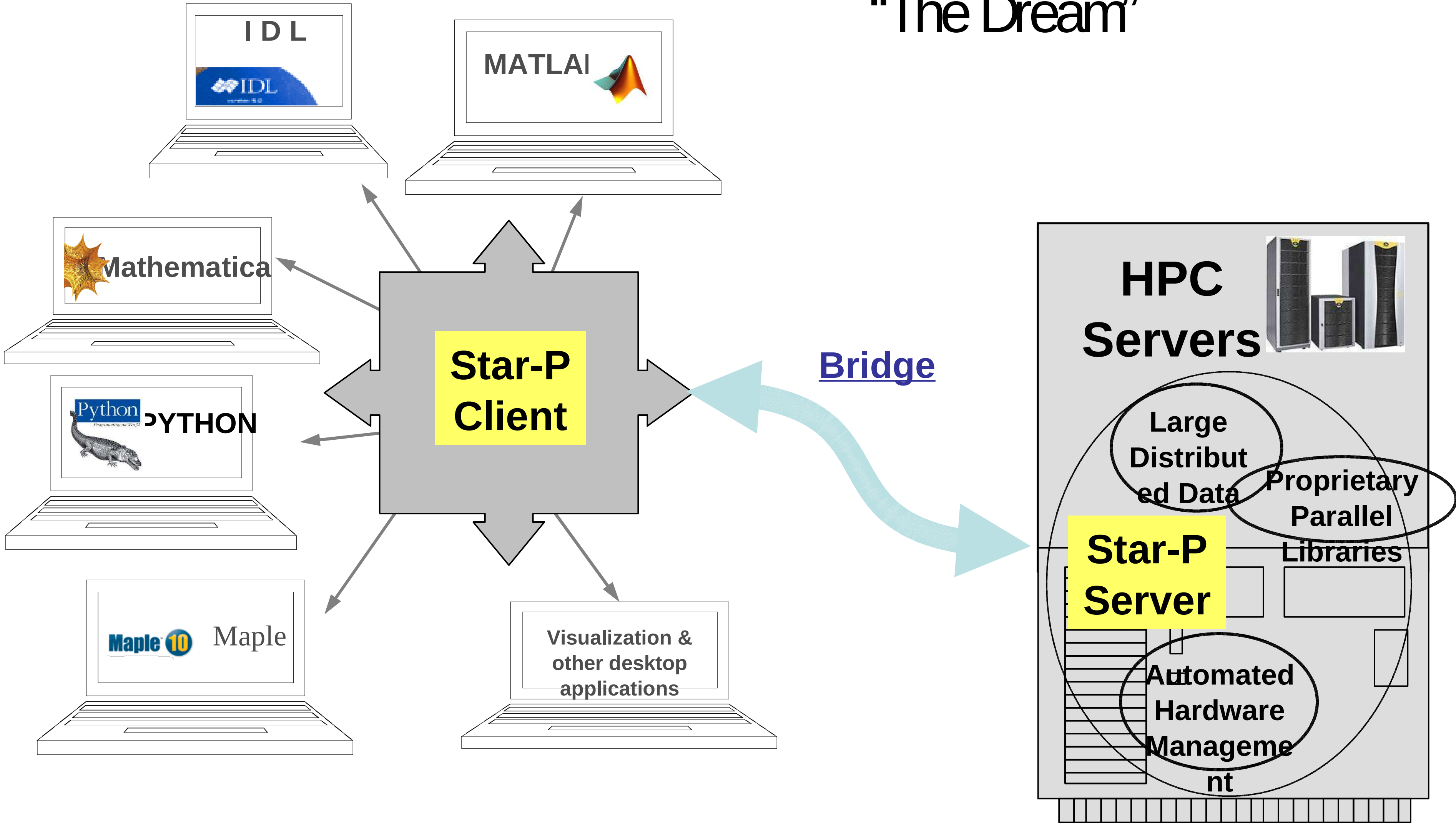


←puzzle pieces working together

Humans interacting online→

Star-P = A Software Platform For Interactive Supercomputing

“The Dream”

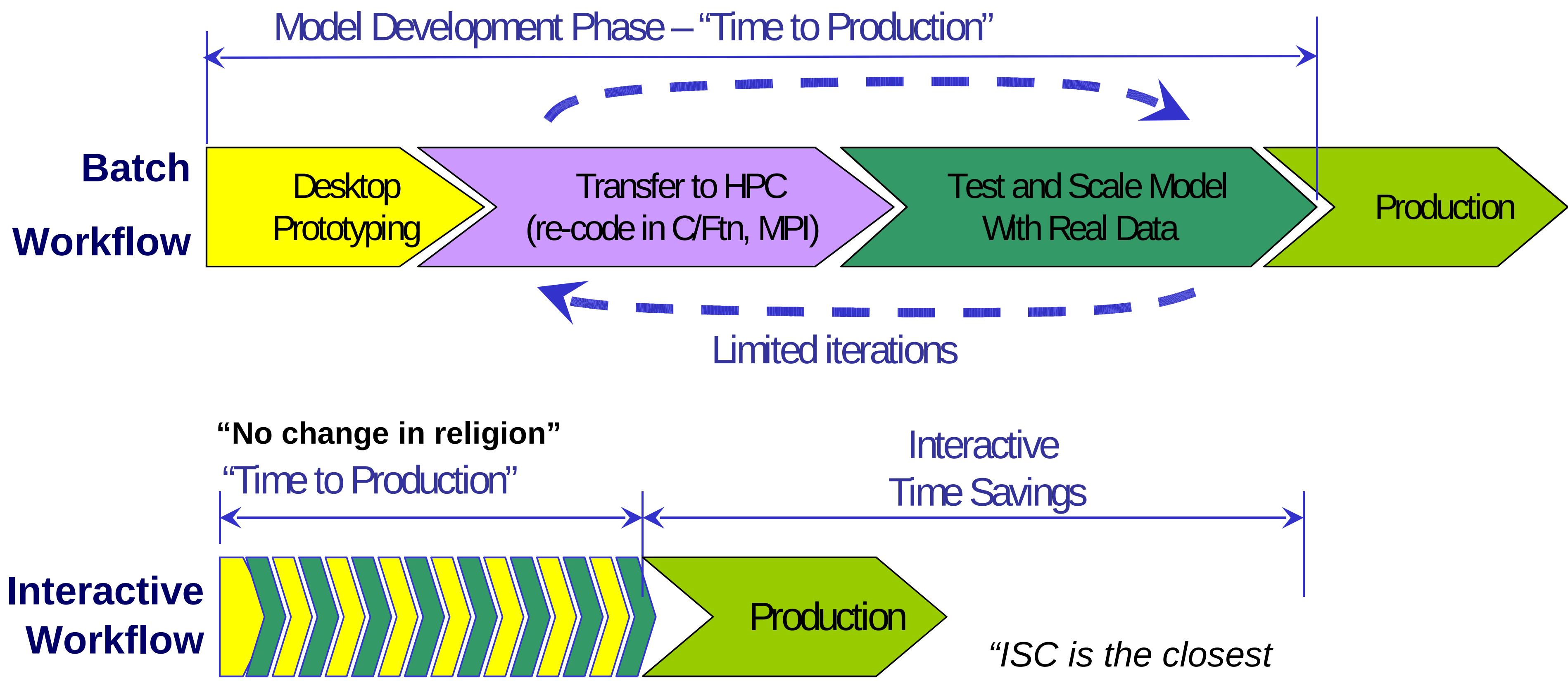


Slide contains trademarks owned by private corporations. Use of logos or registered trademarks does not imply endoresement

Interactive Supercomputing & sgi

INTERACTIVE Fundamentally Alters the Flawed Process

Re-coding takes time, and invariably takes away from model refinement



"ISC is the closest thing I've seen to a killer app."

John Mucci
CEO, SiCortex

Interactive Supercomputing & sgi

High Productivity Design Principles

Rich set of High Performance primitives & tools.

a. Interoperate

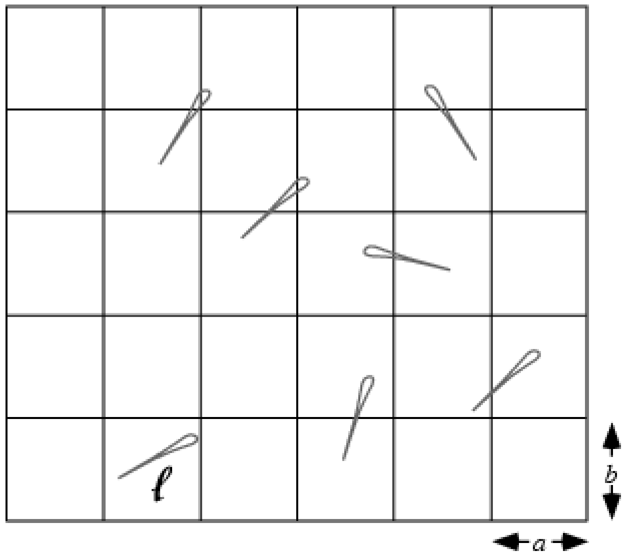
b. Interactive

OK to exploit special-purpose hardware as appropriate (FGPGAs, GPUs)

Do it yourself (in MPI, OpenMP, etc.,) → do it for everyone!

StarP with MATLAB®

The Buffon Needle Problem



$$P(l;a,b)=(2l(a+b)-l^2) / (\pi ab)$$

```
function z=buffon(a,b,l, trials)
    %% Embarassingly Parallel Part
    r=rand(trials,3);
    x =a*r(:,1)+l*cos(2*pi*r(:,3));   % x coord
    y =b*r(:,2)+l*sin(2*pi*r(:,3));   % y coord

    inside = (x >= 0) & (y>=0) & (x <= a) & (y <= b);


    %% Collective Operation (the sum)
    bpi=(2*l*(a+b) - l^2)/ (a*b*(1-sum(inside)/trials));

    %%   Front end
    z=[buffonpi;pi;abs(pi-buffonpi)/pi];
```

buffon(1,1,.5,10000*p)

Interactive Supercomputing & sgi

Star-P Language

MATLAB™, plus

global view (*v.* node-oriented)

Strong bias towards propagation of distributed attribute

*p denotes dimension of distributed array

Overloading of operators

ppeval for task parallelism

Empirical data: typically have to change 10-20 SLOC for MATLAB codes to work in Star-P

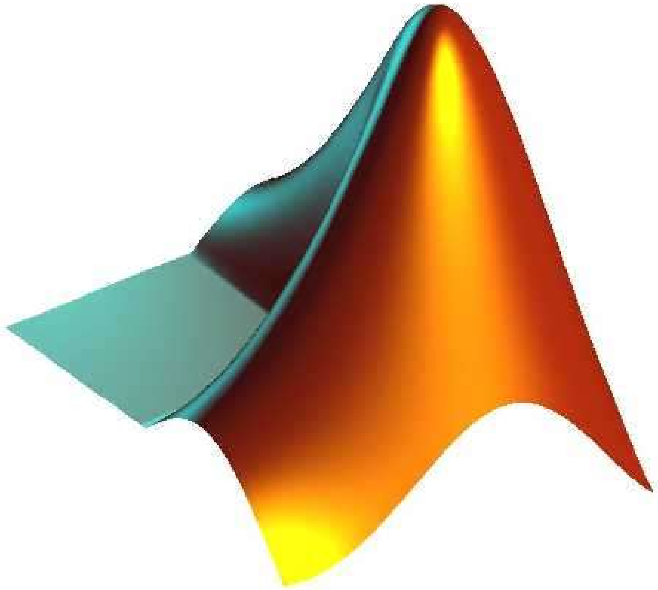
xxx == explicit parallel extension

yyy == parallelism propagated implicitly

```
a = rand(n,n*p);
ppload imagedata a

[nrow ncol] = size(a);
b = ones(nrow,ncol);
c = fft2(a);
d = ifft2(c);

diff = max(max(abs(a-d)));
if (diff > 10*eps)
   sprintf('Error, diff=%f', diff);
end

e = ppeval('sum',a);
e = ppeval('quad','fun',a);
```

Interactive Supercomputing & sgi

It’s still MATLAB!



1. File Editor
2. Profiler
3. Debugger
4. Array Editor
5. Desktop
6. Viz
7. Small Calculations
8. …

Opening Star-P

1. Windows: Hit the Little Button

2. Linux:

   starp <options>

   -a *server_host*

   -t *data_dir_on_server*

   -s *path_to_star-p_on_server*

   -p *number_of_processors*

- Console mode vs desktop mode

Closing Star-P

```
>> quit
```

It’s still MATLAB!

My first Star-P session

```
                         < M A T L A B >

                Copyright 1984-2005 The MathWorks, Inc.

                Version 7.0.4.352 (R14) Service Pack 2

                           January 29, 2005



Connecting to Star-P Server with 4 processes


Star-P Client.

(C) MIT 2002-04.

(C) Interactive Supercomputing, LLC 2004.

All Rights Reserved.

>> 1+1

ans =

     2

>> A=randn(100*p)

 A =

         ddense object: 100p-by-100p
```

My first Star-P session

How many p's?

```
                    < M A T L A B >

                Copyright 1984-2005 The MathWorks, Inc.
                Version 7.0.4.352 (R14) Service Pack 2
                          January 29, 2005


Connecting to Star-P Server with 4 processes


Star-P Client.
(C) MIT 2002-04.
(C) Interactive Supercomputing, LLC 2004.
All Rights Reserved.
>> 1+1
ans =
     2
>> A=randn(100*p)
 A =
          ddense object: 100p-by-100p
```

My first Star-P session

Still MATLAB

How many p’s?

```
                < M A T L A B >
      Copyright 1984-2005 The MathWorks, Inc.
      Version 7.0.4.352 (R14) Service Pack 2
                January 29, 2005

Connecting to Star-P Server with 4 processes


Star-P Client.
(C) MIT 2002-04.
(C) Interactive Supercomputing, LLC 2004.
All Rights Reserved.
>> 1+1
ans =
    2
>> A=randn(100*p)
 A =
        ddense object: 100p-by-100p
```

My first Star-P session

Still MATLAB

How many p's?

Just Checking

```
                    < M A T L A B >
          Copyright 1984-2005 The MathWorks, Inc.
          Version 7.0.4.352 (R14) Service Pack 2
                   January 29, 2005


Connecting to Star-P Server with 4 processes


Star-P Client.
(C) MIT 2002-04.
(C) Interactive Supercomputing, LLC 2004.
All Rights Reserved.
>> 1+1
ans =
    2
>> A=randn(100*p)
 A =
        ddense object: 100p-by-100p
```

My first Star-P session

Still MATLAB

How many p's?

Just Checking

```
                    < M A T L A B >
         Copyright 1984-2005 The MathWorks, Inc.
         Version 7.0.4.352 (R14) Service Pack 2
                    January 29, 2005



Connecting to Star-P Server with 4 processes


Star-P Client.
(C) MIT 2002-04.
(C) Interactive Supercomputing, LLC 2004.
All Rights Reserved.
>> 1+1
ans =
     2
>> A=randn(100*p)
 A =
         ddense object: 100p-by-100p
```

My first Star-P session

```
                    < M A T L A B >
            Copyright 1984-2005 The MathWorks, Inc.
            Version 7.0.4.352 (R14) Service Pack 2
                        January 29, 2005



Connecting to Star-P Server with 4 processes


Star-P Client.
(C) MIT 2002-04.
(C) Interactive Supercomputing, LLC 2004.
                   served.


ans =
     2
>> A=rand(100*p)
 A =
        ddense object: 100p-by-100p
```

100x100 on the server

Data layouts

1. **rand(10*p,10)** row distributed
2. **rand(10,10*p)** column distributed
3. **rand(10*p,10*p)** or **rand(10*p)**

   block cyclic distributed

Data layouts

1. **rand(10*p,10)** row distributed
2. **rand(10,10*p)** column distributed
3. **rand(10*p,10*p)** or **rand(10*p)**

   block cyclic distributed

What is this p anyway?

Data layouts

1. **rand(10*p,10)** row distributed
2. **rand(10,10*p)** column distributed
3. **rand(10*p,10*p)** or **rand(10*p)**

   block cyclic distributed

What is this p anyway?

- World’s dumbest symbolic var?
- Better to tag dimensions than arrays!

Principles

MATLAB language & experience

Minimal code changes

Server has big data

a. Distributed attribute, once established, should be propagated
   → Operators on distributed data should preserve distribution
   → Arrays created via indexing should preserve distribution

b. Data should be moved back to the client only as a last resort, and usually via explicit user direction

c. Some minor behavioral changes OK, as dictated by big data

New Variables / Routines

p

“Symbolic” variable denoting distribution of array dimension

np

a. Number of processors

Small set of added commands (prefixed by “pp”)

a. ppeval (MIMD mode)
b. Data query: ppwhos
c. Data movement: ppload/ppsave, matlab2pp/pp2matlab
d. Performance monitoring: pptic/pptoc

Indexing: Examples

```
a = rand(1000*p);  b = rand(1000*p);


C = a(1:end, 1:end);
D = a(18:23, 47:813);        %all distributed
E = a( : );


F = a(47,18);               % scalar -> local
```

Explicit Data Movement

pp2matlab / matlab2pp

Ideal: Never use pp2matlab

Rather use “display”

Ideal: Never use matlab2pp

Rather use “reshape”

Global Array syntax

```
aa = rand(n,n*p);                       % explicitly parallel with *p
ppload ‘imagedata’ aa    % explicitly parallel with ppload

[nrow ncol] = size(aa);    % implicitly parallel
bb = ones(nrow,ncol);      % “”
cc = fft2(aa);             % “”
dd = ifft2(cc);            % “”

diff = max(max(abs(aa-dd)));
if (diff > 100*eps)
   sprintf(‘Numerical error in fft/ifft, diff=%f’, diff);
end
```

Interactive Supercomputing & sgi

Data Parallel vs Global Array Syntax

Usually used synonymously

Probably unfortunate:

C=A+B is both

Data parallel not GAS

```
for i=1:n, for j=1:n
        c(i,j)=a(i,j)+b(i,j)
end, end
```

Performance Basics

Star-P aimed for big data sizes

a. i.e., bigger than the desktop

“Vectorization” will be important

a. Client/server architecture introduces some latency

b. Communicating with the server in larger chunks preferred

Instrumenting Code

`pptic/pptoc`

a. Usage like `tic/toc`

b. Provides information about client-server traffic and server execution variables (time, counts of key operations)

`PPPROFILING`

```
global PPPROFILING;  PPPROFILING = 1
```

c. Gives information about each client/server call

Large Memory Demo

```
>> np
ans =
   56
>> scale
echo on
n = sqrt(0.8*m/8)
n =
   5.9161e+05
aa = rand(n*p, n*p);
tic ; sum(sum(aa)), toc
ans =
   1.7500e+11
Elapsed time is 260.589829 seconds.
>> whose
Your variables are:
  Name       Size               Bytes            Class
  aa        591607px591607p    2.799991e+12     ddense array
  ans        1x1                8                double array
  m          1x1                8                double array
  n          1x1                8                double array
Grand total is 3.499988e+11 elements using 2.799991e+12 bytes
MATLAB has a total of 3 elements using 24 bytes
```

pptic/pptoc Usage

```
>> a = rand(100);
>> B = rand(100*p);
>> % B is distributed, a is local;  a will get moved
    to the server
>> pptic, C = a+B; pptoc;
Client/server communication info:
    Send msgs/bytes     Recv msgs/bytes      Time spent
  4e+00 / 2.080e+02B  4e+00 / 8.054e+04B   7.032e-01s
Server info:
  execution time on server: 2.621e-02s
  #ppchangedist calls: 0
```

PPPROFILING Usage

```
>> global PPPROFILING ; PPPROFILING = 1
PPPROFILING =
    1
>> a = rand(1000*p)
ppbase_addDense     [   2]
   [1000]
   [1000]
   [   1]
   [   1]
   [   3]
time=0.67036
a =
      ddense object: 1000p-by-1000p
>> b = fft(a)
ppfftw_fft     [1x1 com.isc.starp.ppclient.MatrixID]
   [                          0]
   [                          1]
time=0.30302
ppbase_id2ddata     [6]
time=0.14625
b =
```

Interactive Supercomputing & sgi

Sparse Matrices & Combinatorial Algorithms

Combinatorial Algorithm Design Principle: Do it with a sparse matrix

Graph Operations are well expressed with sparse matrices as the data structure.

Primitives for combinatorial scientific computing.

a. Random-access indexing: `A(i,j)`

b. Neighbor sequencing: `find (A(i,:))`

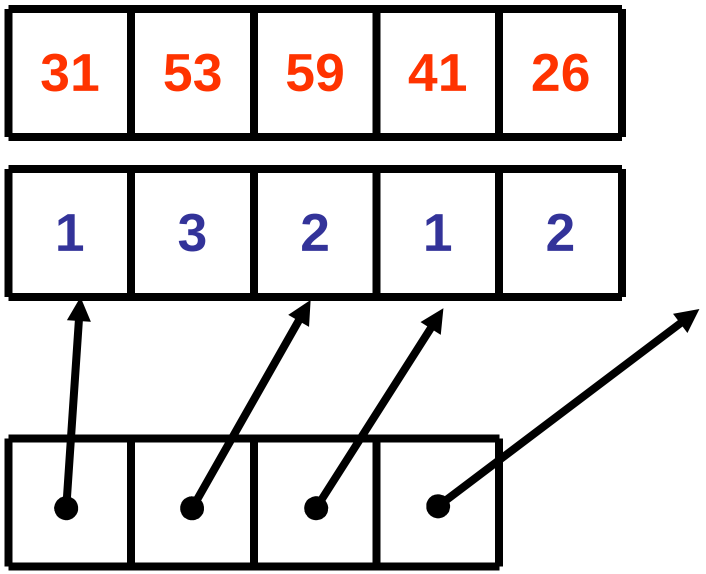
c. Sparse table construction: `sparse (I, J, V)`

d. Matrix * Vector: walking on the graph

Interactive Supercomputing & sgi

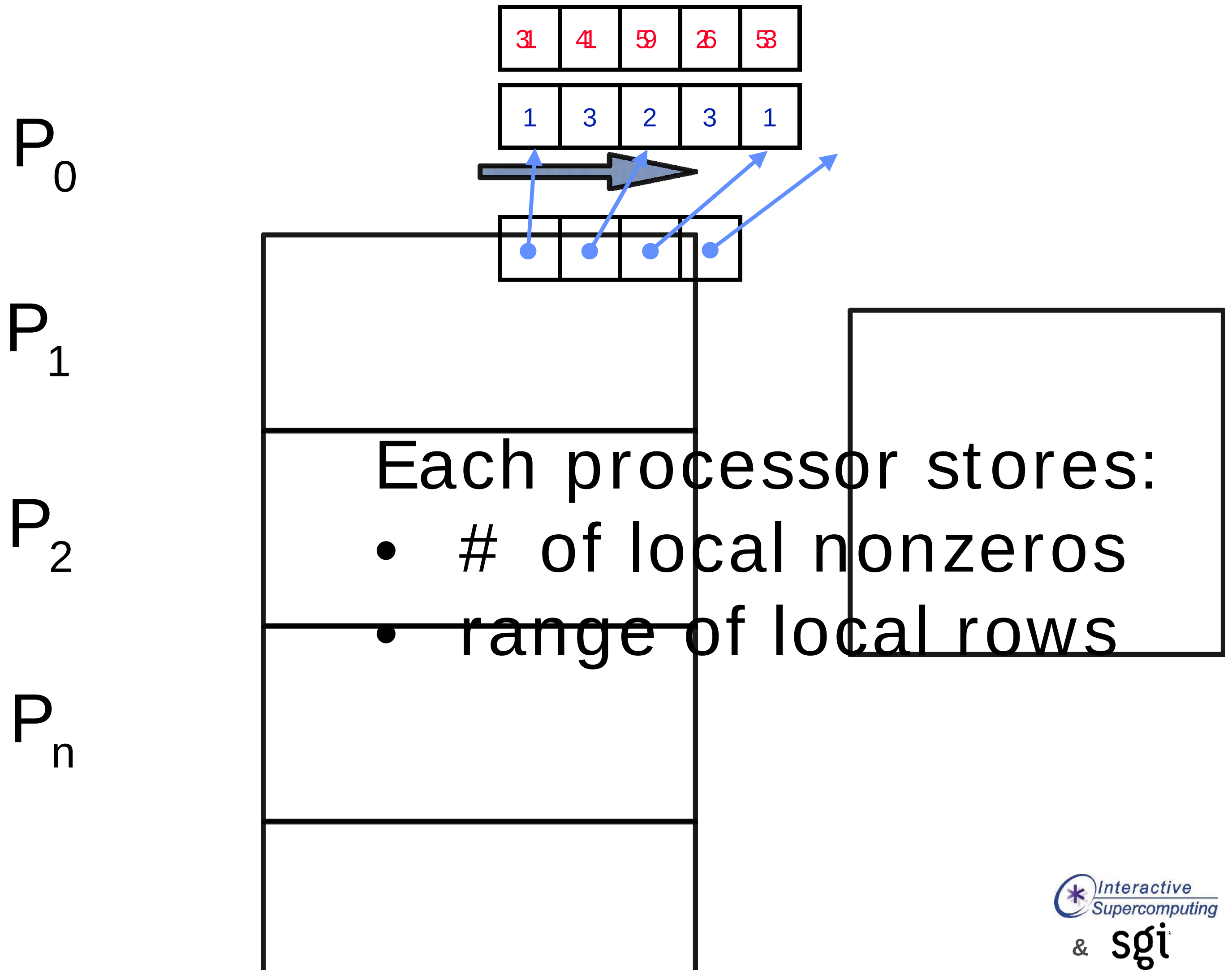
Star-P sparse data structure

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 (2*nzs + nrows) memory

Star-P distributed sparse data structure



Each processor stores:

- # of local nonzeros
- range of local rows

SSCA#2 Graph Theory Benchmark

HPCS High Productivity Computer Systems

Scalable Synthetic Compact Application (SSCA) Benchmarks

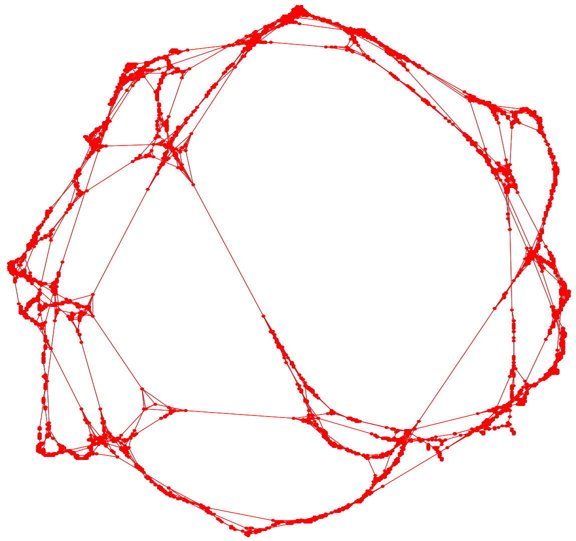
Bioinformatics Optimal Pattern Matching

Graph Theory

Sensor Processing

SSCA#2:- Graph Analysis; stresses memory access; compute-intensive and hard to parallelize.

8192-vertex graph from Kernel 1 plotted with Fiedler coordinates



Interactive Supercomputing & sgi

SSCA#2

Kernel 1: Construct graph data structures

Bulk of time for smaller problems

Kernel 2: Search within large sets

Kernel 3: Subgraph extraction

Kernel 4: Graph clustering

Version does not scale for larger problems

OpenMP Contest:

http://www.openmp.org/drupal/sc05/omp-contest.htm

3. First prize: $1000 plus a 60GB iPod.
4. Second prize: $500 plus a 4GB iPod nano.
5. Third prize: $250 plus a 1GB iPod shuffle

Scalability

Kernels 1 through 3 ran on N=2^{26}

- Previous largest known run is N=2^{21} or 32 times smaller on a Cray MTA-2

- Timings scale reasonably – we played with building the largest sparse matrix we could, until we hit machine limitations!
  - 2xProblem Size → 2xTime
  - 2xProblem Size & 2xProcessor Size → same time

Lines of Code

Lines of executable code (excluding I/O and graphics based on original codes available):

	cSSCA2	The spec	Pthreads
Kernel 1	29	68	256
Kernel 2	12	44	121
Kernel 3	25	91	297
Kernel 4	44	295	241

Expressive Power: SSCA#2 Kernel 3

Star-P (25 SLOC)

```
A = spones(G.edgeWeights{1});
nv = max(size(A));
npar = length(G.edgeWeights);
nstarts = length(starts);
for i = 1:nstarts
    v = starts(i);
    % x will be a vector whose nonzeros
    % are the vertices reached so far
    x = zeros(nv,1);
    x(v) = 1;
    for k = 1:pathlen
        x = A*x;
        x = (x ~= 0);
    end;
    vtxmap = find(x);
    S.edgeWeights{1} = G.edgeWeights{1}…
        (vtxmap,vtxmap);
    for j = 2:npar
        sg = G.edgeWeights{j}(vtxmap,vtxmap)
        if nnz(sg) == 0
            break;
        end;
        S.edgeWeights{j} = sg;
    end;
    S.vtxmap = vtxmap;
    subgraphs{i} = S;
end
```

MATLABmpi (91 SLOC)

```
declareGlobals;


intSubgraphs = subgraphs(G, pathLength, startSetInt);
strSubgraphs = subgraphs(G, pathLength, startSetStr);

%| Finish helping other processors.
if P.Ncpus > 1
  if P.myRank == 0          % if we are the leader
    for unused = 1:P.Ncpus-1
      [src tag] = probeSubgraphs(G, [P.tag.K3.results]);
      [isg ssg] = MPI_Recv(src, tag, P.comm);
      intSubgraphs = [intSubgraphs isg];
      strSubgraphs = [strSubgraphs ssg];
    end
    for dest = 1:P.Ncpus-1
      MPI_Send(dest, P.tag.K3.done, P.comm);
    end
  else
    MPI_Send(0, P.tag.K3.results, P.comm, ...
         intSubgraphs, strSubgraphs);
    [src tag] = probeSubgraphs(G, [P.tag.K3.done]);
    MPI_Recv(src, tag, P.comm);
  end
end


function graphList = subgraphs(G, pathLength, startVPairs)

graphList = [];

% Estimated # of edges in a subgraph. Memory will grow as needed.
estNumSubGEdges = 100;   % depends on cluster size and path length

%--------------------------------------------------------------------
% Find subgraphs.
%--------------------------------------------------------------------

% Loop over vertex pairs in the starting set.
for vertexPair = startVPairs.'
```

```
      % Wait for a response for each request we sent out.
      for unused = 1:numRqst
        [src tag] = probeSubgraphs(G, [P.tag.K3.dataResp]);
        [starts newEdges] = MPI_Recv(src, tag, P.comm);
        subg.edgeWeights{1}(:, starts) = newEdges;
        [newEnds unused] = find(newEdges);
        allNewEnds = [allNewEnds; newEnds];
      end
    end % of if ~P.paral

    % Eliminate any new ends already in the all starts list.
    newStarts = setdiff(allNewEnds.', allStarts);
    allStarts = [allStarts newStarts];

    if ENABLE_PLOT_K3DB
      plotEdges(subg.edgeWeights{1}, startVertex, endVertex, k);
    end % of ENABLE_PLOT_K3DB

    if isempty(newStarts)        % if empty we can quit early.
      break;
    end
   end

  % Append to array of subgraphs.
  graphList = [graphList subg];
end
                function [src, tag] = probeSubgraphs(G, recvTags)


while true
  [ranks tags] = MPI_Probe('*', P.tag.K3.any, P.comm);
  requests = find(tags == P.tag.K3.dataReq);
  for mesg = requests.'
    src = ranks(mesg);
    starts = MPI_Recv(src, P.tag.K3.dataReq, P.comm);
    newEdges = G.edgeWeights{1}(:, starts - P.myBase);
    MPI_Send(src, P.tag.K3.dataResp, P.comm, starts, newEdges);
  end
```

	cSSCA2	executable spec	C/Pthreads/ SIMPLE
Kernel 1	29	68	256
Kernel 2	12	44	121
Kernel 3	25	91	297
Kernel 4	44	295	241

```
startDests = floor((newStarts - 1) / P.myV);
uniqDests = unique(startDests);
for dest = uniqDests
  starts = newStarts(startDests == dest);

  if dest == P.myRank
    newEdges = G.edgeWeights{1}(:, starts - P.myBase);
    subg.edgeWeights{1}(:, starts) = newEdges;
    [allNewEnds unused] = find(newEdges);
  elseif ~isempty(starts)
```

Interactive Supercomputing & sgi

Interactivity!

Did not just build a benchmark: Explored an algorithm space!

Spectral Partitioning based on Parpack was fine for small sizes but not larger.

We played around! We plotted data! We had a good time. ☺ Parallel computing is fun again!

Interactive Supercomputing

No “change in religion”

a. Use familiar tools

b. Desktop, interactive

5-10x manpower savings by transforming workflow

a. Enables rapid (and more frequent) iteration

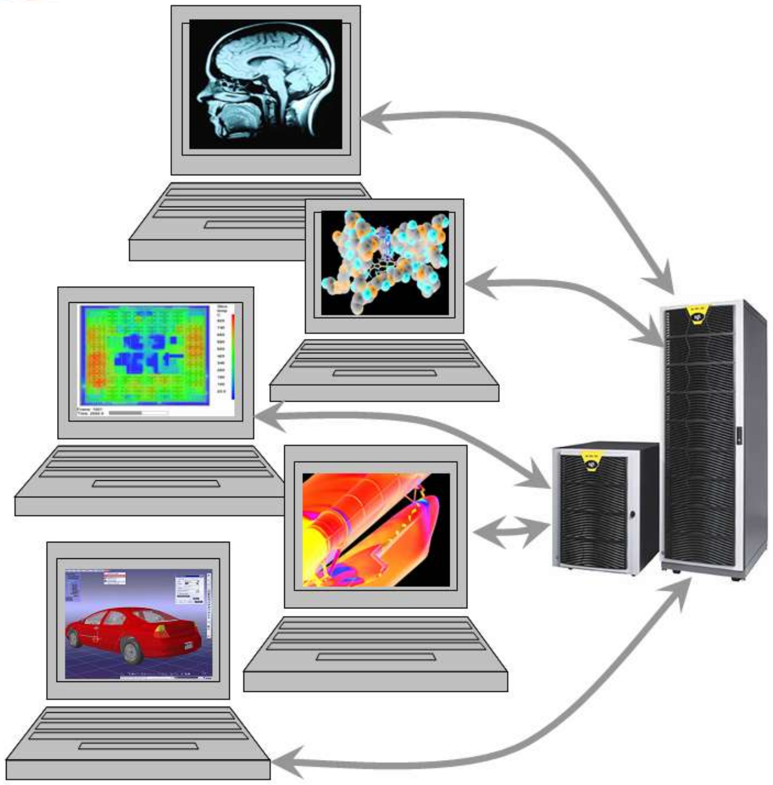
b. Drives better conclusions, decisions, products

Improves “Time to Production”

a. 50% reductions in calendar time

b. Improves time to market

c. Increases profits



"In computing with humans, response time is everything….One's likelihood of getting the science right falls quickly as one loses the ability to steer the computation on a human time scale."

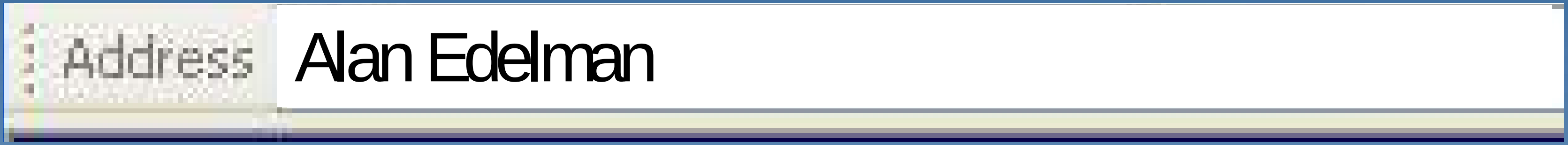
Prof. Nick Trefethen
Oxford University

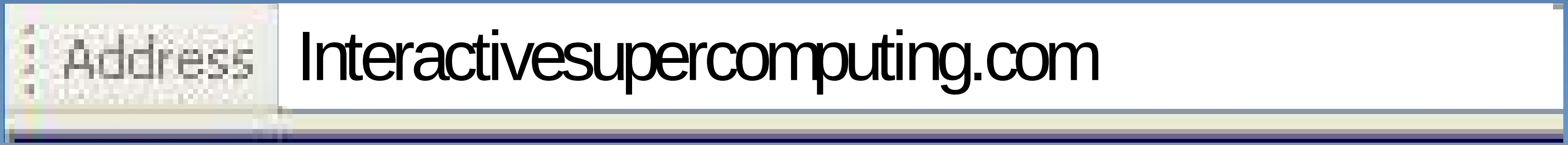
Interactive Supercomputing & sgi

Interactive Supercomputing

Delivering Interactive Parallel Computing Power to the Desktop

sgi


Address Alan Edelman


Address Interactivesupercomputing.com

Contact:Alan Edelman

Interactive Supercomputing • 135 Beaver Street, FL 2 • Waltham, MA 02452

781.398-0010 • edelman@interactivesupercomputing.com • www.interactivesupercomputing.com