An Interactive Approach to Parallel Computing Algorithms with Star-P

Alan Edelman MIT (& ISC)
Parallel Computing Arts

Message Passing:

Batch Processing:
Coding, Modeling, & Debugging

Punch Cards (textile loom 1840)

Noble perfected arts: what’s next for productivity?

The King’s Messenger
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"

Star-P Client

HPC Servers

Large Distributed Data

Proprietary Parallel Libraries

Star-P Server

Automated Hardware Management

Maple

Mathematica

Python

IDL

MATLAB

Visualization & other desktop applications

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

Building a model takes time and is a crucial part of the development process. However, re-coding takes even more time and can significantly delay the refinement of the model. This is where the interactive workflow comes in.

Interactive Workflow:

- **Desktop Prototyping**
- **Transfer to HPC** (re-code in C/Ftn, MPI)
- **Test and Scale Model With Real Data**
- **Production**

Interactive workflow offers significant time savings compared to batch workflow.

"I've seen the closest thing I've seen to a killer app."  
John Mucci  
CEO, SiCortex
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!
The Buffon Needle Problem

\[ P(l; a, b) = \frac{(2l(a+b) - l^2)}{\pi ab} \]

```matlab
function z = buffon(a, b, l, trials)
    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; abs(pi - buffonpi) / pi];
end
```

```matlab
buffon(1, 1, 5, 10000*pi)
```
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

---

```matlab
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);
```
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

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 =
   dense object: 100p-by-100p
My first Star-P session

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 =
    dense object: 100p-by-100p
My first Star-P session

Still MATLAB

How many p’s?

MATLAB>

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 =

dense object: 100p-by-100p
My first Star-P session

Still MATLAB

How many p’s?

Just Checking

MATLAB>

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 =

dense object: 100p-by-100p
My first Star-P session

Still MATLAB

How many p’s?

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 =
    dense object: 100p-by-100p

Just Checking
My first Star-P session

Connecting to Star-P Server with 4 processes

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

100x100 on the server

ans =
2

>> A=rand(100*p)
A =

dense object: 100p-by-100p
Data layouts

1. \( \text{rand}(10*p, 10) \) row distributed
2. \( \text{rand}(10, 10*p) \) column distributed
3. \( \text{rand}(10*p, 10*p) \) or \( \text{rand}(10*p) \)
   block cyclic distributed
Data layouts

1. \texttt{rand(10*p,10)} row distributed
2. \texttt{rand(10,10*p)} column distributed
3. \texttt{rand(10*p,10*p)} or \texttt{rand(10*p)}
   block cyclic distributed

What is this p anyway?
1. \( \text{rand}(10\times p, 10) \) row distributed
2. \( \text{rand}(10, 10\times p) \) column distributed
3. \( \text{rand}(10\times p, 10\times p) \) or \( \text{rand}(10\times p) \)
   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. \( \text{ppeval} \) (MIMD mode)

b. Data query: \( \text{ppwhos} \)

c. Data movement: \( \text{ppload/ppsave, matlab2pp/pp2matlab} \)

d. Performance monitoring: \( \text{pptic/pptoc} \)
Indexing: Examples

\[ a = \text{rand}(1000*p); \quad b = \text{rand}(1000*p); \]

\[ C = a(1:\text{end}, 1:\text{end}); \]
\[ D = a(18:23, 47:813); \quad \% \text{all distributed} \]
\[ E = a(:\text{end}); \]

\[ F = a(47,18); \quad \% \text{scalar} \rightarrow \text{local} \]
Explicit Data Movement

pp2matlab / matlab2pp

Ideal: Never use pp2matlab
Rather use “display”

Ideal: Never use matlab2pp
Rather use “reshape”
Global Array syntax

\[ a_a = \text{rand}(n,n*p); \quad \% \text{explicitly parallel with } *p \]

\text{ppload} \text{"imagedata"} a_a \quad \% \text{explicitly parallel with } \text{ppload}

\[ [\text{nrow ncol}] = \text{size}(a_a); \quad \% \text{implicitly parallel} \]

\text{bb} = \text{ones}(\text{nrow},\text{ncol}); \quad \% \text{""} \]

\text{cc} = \text{fft2}(a_a); \quad \% \text{""} \]

\text{dd} = \text{ifft2}(\text{cc}); \quad \% \text{""} \]

\text{diff} = \text{max(max(abs(a_a-dd))))};

\text{if } (\text{diff} > 100*\text{eps})

\hspace{1cm} \text{printf('Numerical error in fft/ifft, diff=\%f', diff)};

\text{end}
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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>aa</td>
<td>591607px591607p</td>
<td>2.799991e+12</td>
<td>ddense array</td>
</tr>
<tr>
<td>ans</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>m</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
<tr>
<td>n</td>
<td>1x1</td>
<td>8</td>
<td>double array</td>
</tr>
</tbody>
</table>

Grand total is 3.499988e+11 elements using 2.799991e+12 bytes
MATLAB has a total of 3 elements using 24 bytes
$\texttt{pptic/pptoc Usage}$

\begin{verbatim}
>> 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
\end{verbatim}
>> global PPPROFILING ; PPPROFILING = 1
PPROFILING =
    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.starppclient.MatrixID]
    [0]
    [1]
time=0.30302
ppbase_id2ddata [6]
time=0.14625
b =
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: \( \text{find} \ (A(i,:)) \)

c. Sparse table construction: \( \text{sparse} \ (I, J, V) \)

d. Matrix * Vector: walking on the graph
Star-P sparse data structure

- **Full:**
  - 2-dimensional array of real or complex numbers
  - \((\text{nrows} \times \text{ncols})\) memory

- **Sparse:**
  - compressed row storage
  - about \((2 \times \text{nzs} + \text{nrows})\) memory
Star-P distributed sparse data structure

Each processor stores:
- # of local nonzeros
- range of local rows
SSCA#2 Graph Theory Benchmark

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.
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!
  - \( 2 \times \text{Problem Size} \rightarrow 2 \times \text{Time} \)
  - \( 2 \times \text{Problem Size} \& 2 \times \text{Processor Size} \rightarrow \text{same time} \)
Lines of executable code (excluding I/O and graphics based on original codes available):

<table>
<thead>
<tr>
<th></th>
<th>cSSCA2</th>
<th>The spec</th>
<th>Pthreads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel 1</td>
<td>29</td>
<td>68</td>
<td>256</td>
</tr>
<tr>
<td>Kernel 2</td>
<td>12</td>
<td>44</td>
<td>121</td>
</tr>
<tr>
<td>Kernel 3</td>
<td>25</td>
<td>91</td>
<td>297</td>
</tr>
<tr>
<td>Kernel 4</td>
<td>44</td>
<td>295</td>
<td>241</td>
</tr>
</tbody>
</table>
```
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 = zeros(nv,1);
    for k = 1:pathlen
        x = A*x;
    end;
    vtxmap = find(x);
    S.edgeWeights{1} = G.edgeWeights{1}...
        (vtxmap,vtxmap);
    for j = 2:npar
        sg = G.edgeWeights{j}(vtxmap,vtxmap);
    end;
    S.vtxmap = vtxmap;
    subgraphs{i} = S;
end
```

MATLABmpi (91 SLOC)

```
if isemtpy(msg)
    msg = find(ismember(tags,recvTags));
    if msg = = request{'}
        src = ranks(msg);
        request = find(src = = P.tag.K3.dataResp);
        for msg = = requests'
            srch = P.tag.K3.dataResp;
            sg = G.edgeWeights{1}(vtxmap,vtxmap);
            subgraphs = find(vertLabels{1},start = = dest);
            subg.edgeWeights{1} = sg;
        end;
    end
end
```
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