

MATLAB*P: Architecture

Ron Choy, Alan Edelman

Laboratory for Computer Science

MIT

Outline

- The “p” is for parallel
- MATLAB is what people want

Supercomputing in 2003

- The impact of the Earth Simulator
- The impact of the internet bubble
- The rise of clusters
- The few big players left

The Rise of Clusters

- Cheap & everyone seems to have one



Some MIT Clusters

Site	Name	Processors/Nodes	Type	Memory	Interconnect	Purpose	Built/Hardware updated	Maintained
LCS Computer Architecture Group	cagfarm	100/50	1GHz to 2.2GHz P4 Xeons	2GB x 50 = 100GB	Gigabit Ethernet	Research	??	Tech Square Inc.
Parsons Laboratory (Building 48)	dair.mit.edu	84/42	Pentium 4, Xeon	84GB	Myrinet and Ethernet	Environmental data assimilation	2002	Internal to project + Aspen Systems remote support
LCS Computer Graphics Group - 1	photon	64/32/128 (HyperThreading)	Intel Xeon 2.4GHz	80GB	Gigabit Ethernet	Research	??	Department sysadmin
LCS Computer Graphics Group - 2	tesseract	64/32	Intel P4 400MHz FSB	20GB	Fast Ethernet	Research	??	Department sysadmin
Materials Science, Building 13	lambic.mit.edu	60/30	Athlon 1900/Xeon 2.8/Alpha 833	2GB x 30 = 60GB	Ethernet/ Myrinet	First-principles calculations in materials/Teaching SMA5107	2000	Postdocs and Students
Earth Atmospheric and Planetary Sciences (E54)	MITGCM Cluster	56/44	Pentium 4, 533MHz FSB	22GB (1066MHz RDRAM)	Myrinet and Ethernet	Climate Science	2002	Tech Square Inc.
CTP Lattice Group/ LNS	lqcd.lns.mit.edu	48/12	Alpha EV67 666MHz	12GB	Myrinet	Research in LQCD	1998	Andrew Pochinsky
Laboratory for Nuclear Science	?	40/20	Pentium II @ 400 MHz	512MB x 20 = 10 GB	Fast Ethernet	Scientific Computing for LNS community	1998	J. Maynard Gelinas
Applied Mathematics	heavyside	32/16	Athlon MP 2000+	1GB x 16 = 16 GB	Gigabit Ethernet	Research in materials simulations, granular flows, electroosmosis, percolation, fractal growth, and other physical systems.	Sept 2002	Department sysadmin
MIT Nuclear Engineering Department	Echelon	30/30	Athlon 1700 1.1GHz OC 1.5GHz	0.5GB * 30 = 15GB PC2100	Fast Ethernet	Monte Carlo N Particle code for reactor criticality	2002	Graduate Student
Laboratory for Nuclear Science	?	20/20	Athlon 900 MHz	512MB x 20 = 10GB	Fast Ethernet	Scientific Computing for LNS community	2001	J. Maynard Gelinas
LCS Applied Computing Group	beowulf.lcs	16/8	Athlon MP 1800+	1GB x 8 = 8GB	Fast Ethernet	Research and Teaching 18.337/6.338/SMA5505	Jan 2002	Ron Choy
LCS Computer Graphics Group - 3	??	8	MIPS R10000/R12000 500MHz	6GB	Fast Ethernet	Research	??	Department sysadmin
LIDS Optical & Satellite Networking Group	throback.mit.edu	8/4	Athlon MP 2100+	2GB x 4 = 8GB	Fast Ethernet	Research in Linear Programming, Optimization, Convex analysis, Network simulations	Nov 2002	Departmental sysadmin/Grad Students

Rise of Clusters

- A Beowulf is in the top 5 of the top500 list!
- Increased computing power does not necessarily translate to increased productivity
- Someone has to program these parallel machines, and this can be hard

Classes of interesting problems

- Large collection of small problems – easy, usually involves running a large number of serial programs – embarrassingly parallel
- Large problems – might not even fit into the memory of a machine. Much harder since it usually involves communication between processes

Parallel programming (1990 – now)

- Dominant model: MPI (Message Passing Interface)
- Low level control of communications between processes
- Send, Recv, Scatter, Gather ...

MPI

- It is not an easy way to do parallel programming.
- Error prone – it is hard to get complex low level communications right
- Hard to debug – MPI programs tend to die with obscure error messages. There is also no good debugger for MPI.

MPI (2)

- Show some MPI code here ...

The result?

- A lot of researchers in the sciences need the power of parallel computing, but do not have the expertise to tackle the programming
- Time saved by using parallel computers is offset by the additional time needed to program them

The result? (2)

- A high level tool is needed to hide the complexity away from the users of parallel computers

Let's go back to the
serial world
for a moment

Outline

- The “p” is for parallel
- **MATLAB** is what people want

What do people want?

- Real applications programmers care less about “p-fold speedups”
- Ease of programming, debugging, correct answer matters much more

MATLAB

- MATrix LABoratory
- Started out as an interactive frontend to LINPACK and EISPACK
- Has since grown into a powerful computing environment with ‘Toolboxes’ ranging from neural networks to computational finance to image processing.
- Many current users know nothing about linear algebra

MATLAB ‘language’

- MATLAB has a C-like scripting language, with HPF-like array indexing
- Interpreted language
- A wide range of linear algebra routines
- Very popular in the scientific community as a prototyping tool – easier to use than C/FORTRAN

MATLAB 'language' (2)

- MATLAB makes use of ATLAS (optimized BLAS) for basic linear algebra routines
- Still, performance is not as good as C/FORTRAN, especially for loops
- However MATLAB is catching up with its release 13 – (just-in-time compiling?)

Why people like MATLAB?

- Convenience, not performance!

Wouldn't it be nice to mix
the convenience of MATLAB
with
parallel computing?

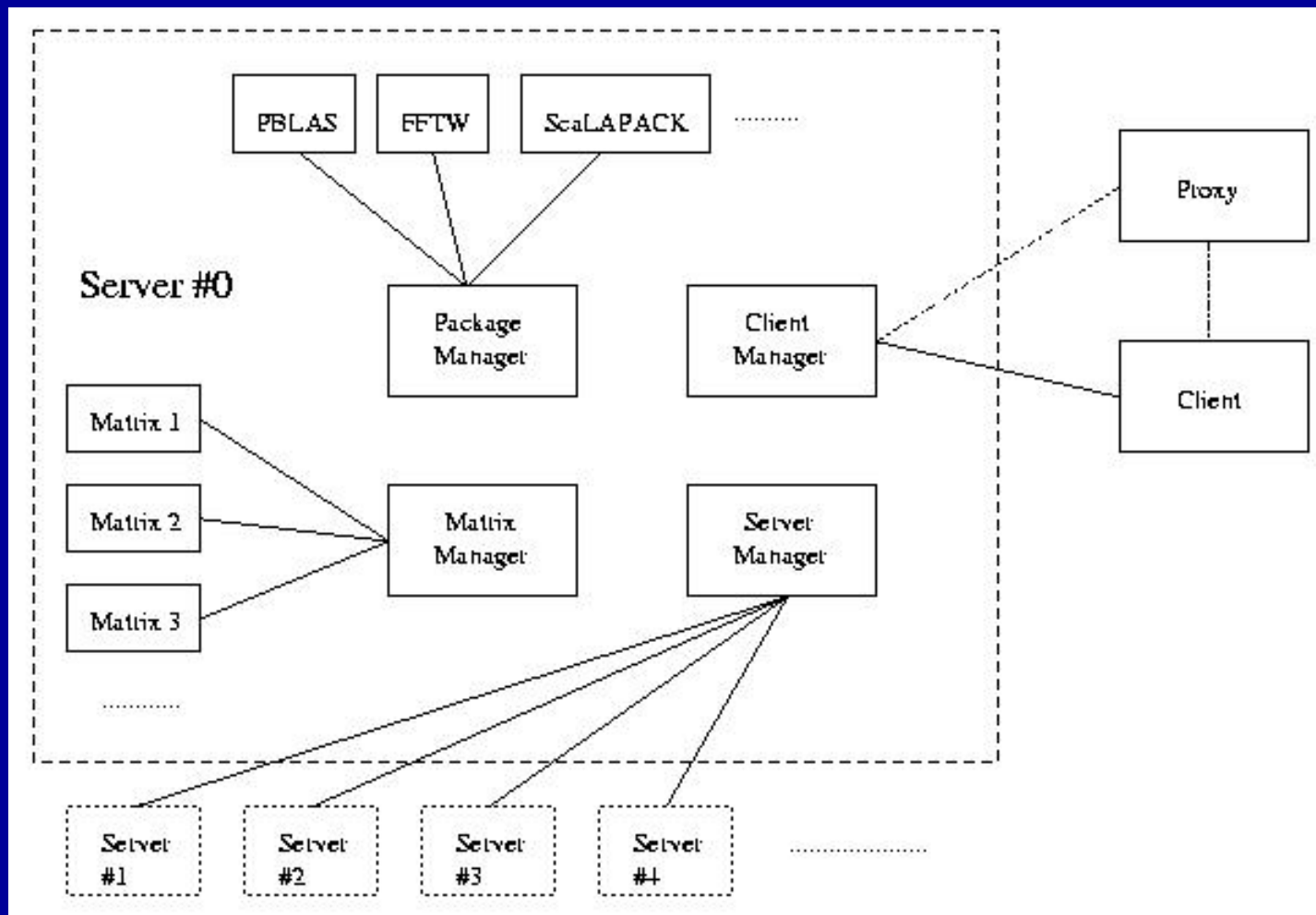
MATLAB*P

- Project started by P. Husbands, who is now at NERSC
- Goal: To provide a user-friendly tool for parallel computing
- Uses MATLAB as a front end to a parallel linear algebra server

MATLAB*P (2)

- Server leverages on popular parallel numerical libraries:
 - ScaLAPACK
 - FFTW
 - PARPACK
 - ...

Structure of the System



Structure of the system (2)

- Client (C++)
 - Attaches to MATLAB through MEX interface
 - Relays commands to server and process returns
- Server (C++/MPI)
 - Manipulate/process data (matrices)
 - Perform the actual parallel computation by calling the appropriate library routines

Functionalities provided

- PBLAS, ScaLAPACK
 - solve, eig, svd, chol, matmul, +, -, ...
- FFTW
 - 1D and 2D FFT
- PARPACK
 - eigs, svds
- Support provided for s,d,c,z

Focus

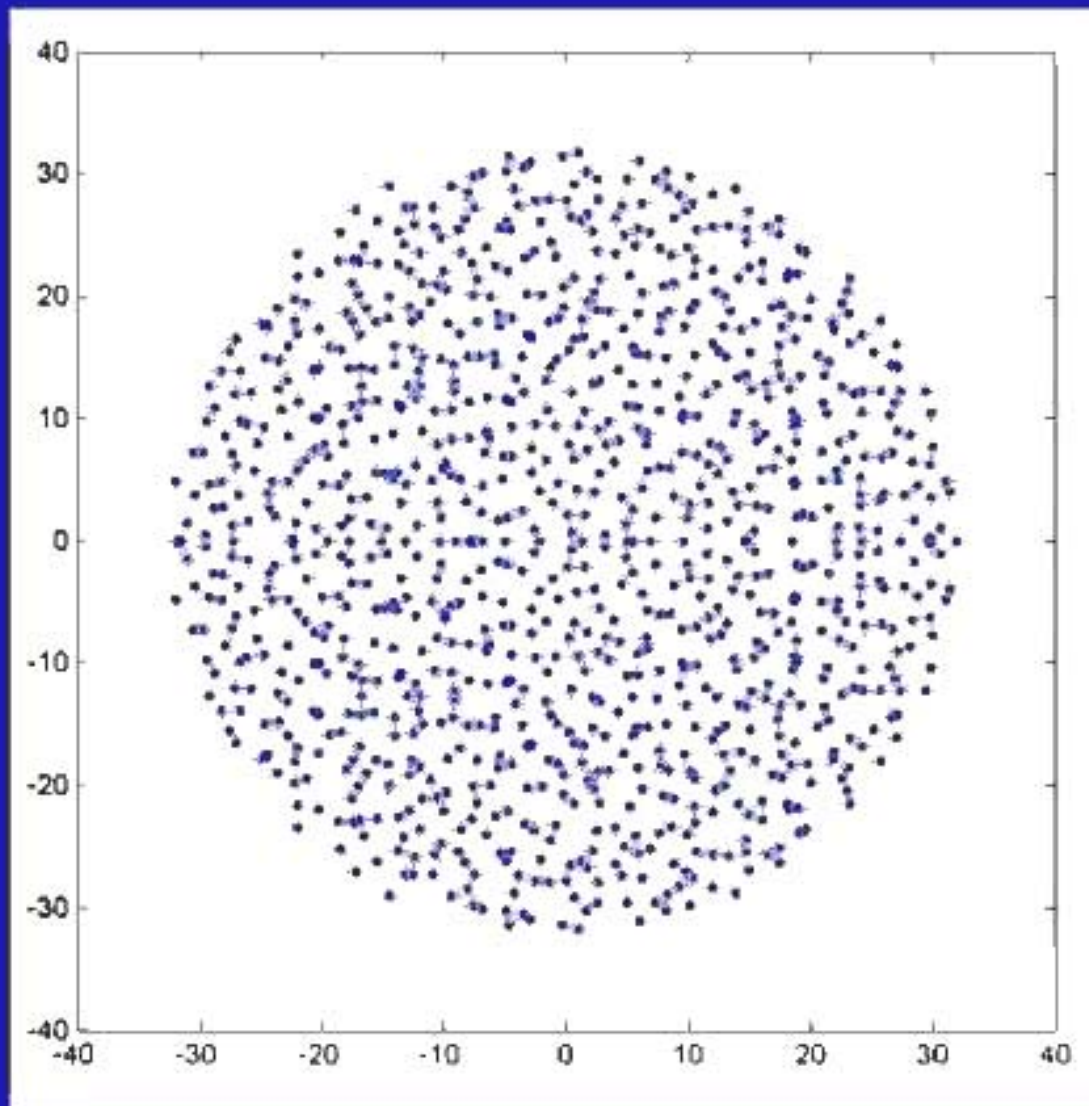
- Require minimal learning on user's part
- Reuse of existing scripts
- Mimic MATLAB behaviour
- Data stay on backend until explicitly retrieved by user
- Extendable backend

What is *p?

- MATLAB*P creates a new MATLAB class called layoot
- By providing overloaded functions for this new class, parallelism is achieved transparently
- *p is layoot(1)

Example

- `A = randn(1024*p,1024*p);`
- `E = eig(A);`
- `e = pp2matlab(E);`
- `plot(e,'*');`



Example 2

- `H = hilb(n)` `% H = hilb(8000*p)`
 - `J = 1:n;`
 - `J = J(ones(n,1),:);`
 - `I = J';`
 - `E = ones(n,n);`
 - `H = E./(I+J-1);`

Built-in MATLAB functions

- Hilb is a built-in MATLAB function that get parallelized automatically
- Another example of a built-in MATLAB function that works is cgs
- We do not know how many of them works

Where is the data?

- One often-asked question in parallel computing is: where is the data residing?
- MATLAB*P supports 3 data distributions:
 - $A = \text{randn}(m * p, n)$ – row distribution
 - $B = \text{randn}(m, n * p)$ – column distribution
 - $C = \text{randn}(m * p, n * p)$ – 2D block cyclic distribution

Flow of a MATLAB*P call

- User type $A = \text{randn}(256*p)$ in MATLAB
- The overloaded `randn` in `layoot` class is called
- The overloaded method calls MATLAB*P client with the arguments
‘ppbase_addDense’, 256, 3 (for cyclic dist)
- Client relays the command to server

Flow of a MATLAB*P call (2)

- Server receives the command and pass it on to PackageManager
- PackageManager finds appropriate function in package PPBase, makes the call
- A 2D block cyclic distributed randn matrix is created
- Server pass matrix handle to client, which passes it to MATLAB. Handle is stored in A.

MultiMATLAB mode

- Similar to MultiMATLAB from Cornell
- Start up a MATLAB process on each node, and run MATLAB scripts on them in parallel
- Interesting feature is that the data can come from the 'regular' mode of MATLAB*P

Example 3

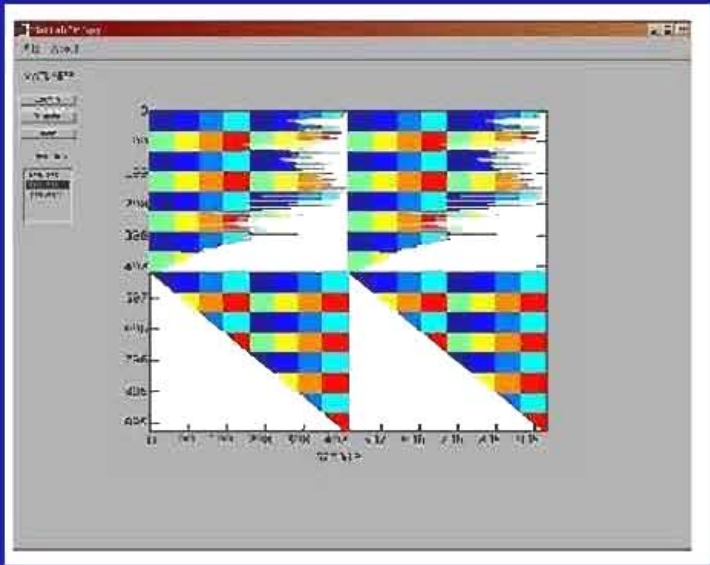
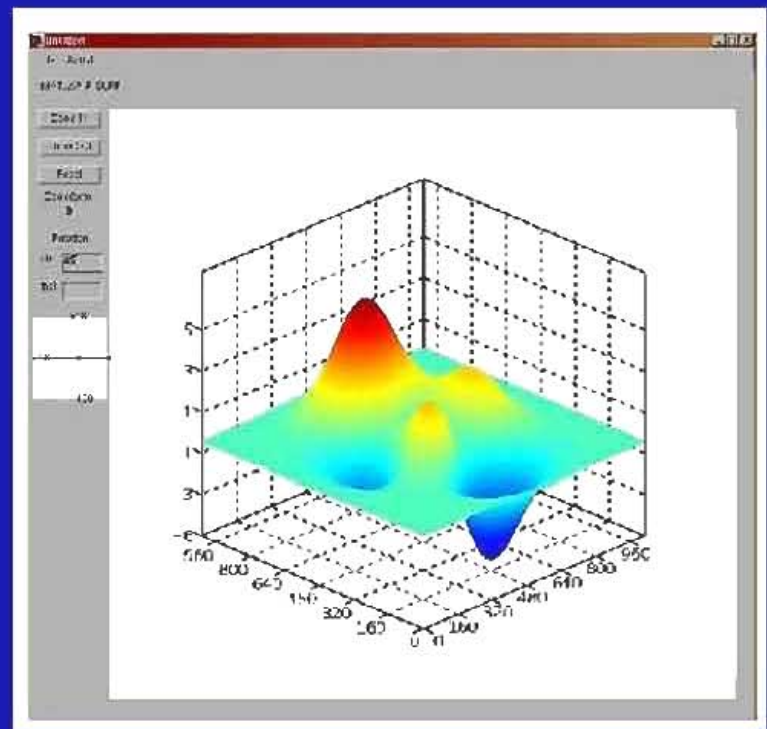
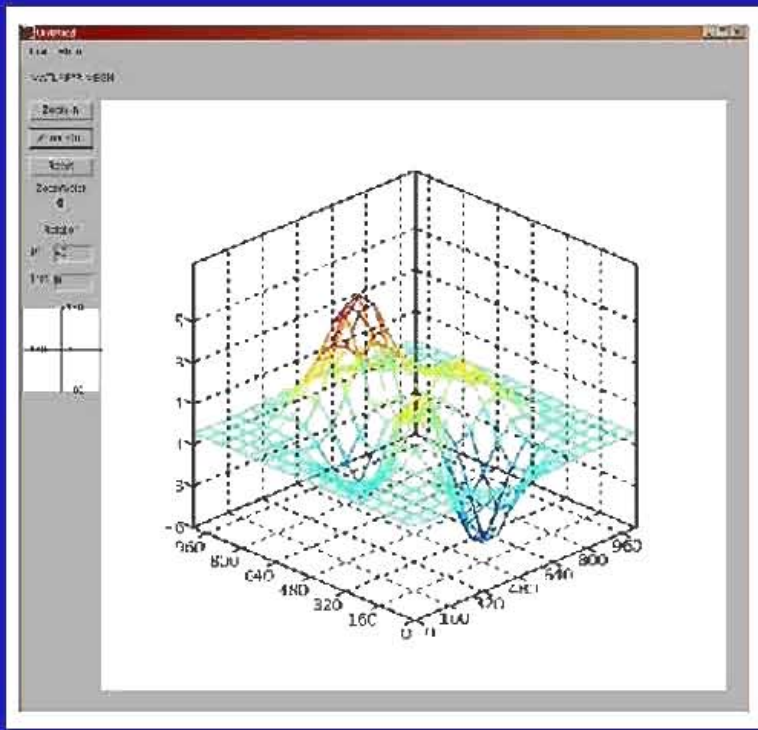
- `d = randn(10000*p,1);`
- `a = 1:9999*p; % created in 'regular' mode`
- `z = mm('chi2rnd',a);`
- `e = stebz(d,z)`

Types of parallel problems

- Collection of small problems
 - MultiMATLAB mode
- Large problems
 - ‘Regular mode’ – ScaLAPACK, FFTW, ...
- We even allow a mixture of the two modes to give additional flexibility

Visualization package

- A term project done by a group of students in a parallel computing class at MIT
- Provides the equivalent of mesh, surf and spy to distributed matrices
- Visualization of very large matrix is now possible in MATLAB



Why parallel MATLAB?

- Why not Maple/Mathematica?
 - MATLAB*P is not really tied to MATLAB – the server does not understand a single line of MATLAB
 - Interfaces could be written for other software ‘easily’
 - We choose MATLAB because it is widely used and easy to use – we want to inherit this property
 - Parallelizing maple/mathematica is a much different problem than parallelizing matlab and probably would have a smaller audience

Why is this interesting?

- Overhead is incurred!
 - Reduction in programming time compensates the overhead
 - Makes large numerical experiments easy

More

- Kinds of matlab users: 0 matlabs, 1 matlab, many matlabs
- Processor maps
- Dynamic addition of processors
- Matlab*p on a grid