Distributed Computing in the Engineering Workflow

Loren Dean
Director of Engineering, MATLAB Products
The MathWorks
Agenda

An important trend impacting the Engineering workflow

- Task parallel applications
- Data parallel applications
Market trend: from single processor to grids

Single processor  Multi-core  Multiprocessor  Clusters  Grids
Why is this important?

- Technical computing
  - Modeling and analysis involves numerous runs
    - Monte carlo or similar applications very common
    - Complexity of algorithms causes longer execution times
  - Data sets are increasing in size

- Model-based design
  - Simulation is done prior to real-world implementation
    - Many scenarios tested
    - Optimal solutions can be found earlier
  - Simulations are growing in complexity and size
    - Simulation time increases
    - Rsim and other targets are only part of the solution
MATLAB addresses the market trend

Single processor  Multi-core  Multiprocessor  Clusters  Grids

MATLAB in R2006b

MathWorks
Aerospace and Defense Conference ’07
MATLAB addresses the market trend

Single processor | Multi-core | Multiprocessor | Clusters | Grids

MATLAB in R2007a

MathWorks
Aerospace and Defense Conference ’07
MATLAB to Distributed Computing

Single processor  Multi-core  Multiprocessor  Clusters  Grids

Distributed Computing Toolbox in R2007a  MATLAB Distributed Computing Engine

MathWorks
Aerospace and Defense Conference '07
What is happening because of this trend?

- Interactive programming - re-use of existing applications
- Distributed/parallel engineering software tools becoming available
- Schedulers and Operating Systems providing foundation services
- Low-cost hardware

ScaLAPACK

4xShuttle

$4,000
Skills required for distributed computing today

- **Cluster Type**
  - Individual (1, Multicore, SMP)
  - Workgroup (<8, 8, 16)
  - Departmental (16, 32)
  - Divisional (64, 96, 128)
  - Capability, Enterprise (128+)

- **User Type**
  - HPC user
  - "Active" user
  - "Passive", Never ever user

- **Skills**
  - "Active" user
  - "Passive", Never ever user

- **IT Specialization**
  - Engineering / Science Degree
  - CS Degree

MathWorks
Aerospace and Defense Conference '07
Skills that would be preferred

- **Cluster Type**
  - Capability, Enterprise (128+)
  - Divisional (64, 96, 128)
  - Departmental (16, 32)
  - Workgroup (<8, 8, 16)
  - Individual (1, Multicore, SMP)

- **User Type**
  - HPC user
  - “Active” user
  - “Passive”, Never ever user

- **Skills**
  - CS Degree
  - Engineering / Science Degree
  - IT
Distributed computing solution

Client Machine

MATLAB

SIMULINK

Toolboxes

Blocksets

Distributed Computing Toolbox

Compute cluster

MATLAB Distributed Computing Engine

Scheduler

Worker

Worker

Worker
Agenda

- An important trend impacting the Engineering workflow
  - Task parallel applications
  - Data parallel applications
Multiple independent problems
Task parallel applications
Example: Land Classification

- National Land Cover Dataset (NLCD) from U.S. Geological Survey – 30GB
- “Where are wetlands, forests etc concentrated?”
- “How does the distribution compare with other datasets?”
From sequential to distributed: MATLAB

```matlab
function results = main(var1, var2)

nSims = 1000;
out = cell(1, nSims);

for ii = 1:nSims
    out{ii} = myFunction(ii, var1, var2);
end

results = postprocessing(out);
```
function results = main(var1, var2)

jm = findResource('scheduler', 'type', 'jobmanager');

job = createJob(jm, ...
    'FileDependencies', 'myFunction.m', ...
    'PathDependencies', [{'myPath\myFolder\data'}]);

nSims = 1000;
out = cell(1, nSims);

for ii = 1:nSims
    createTask(job, @myFunction, 1, {ii, var1, var2});
end

results = postprocessing(out);
1. Divide the Monte Carlo simulations such that each processor executes a full Simulink simulation or RSIM target. 
   Eg., one simulation per altitude

2. Create a Task Function that uses MATLAB commands to call the Simulink model you want to execute
Agenda

- A little history and context setting
- Task parallel solutions

Data parallel solutions
Data parallel applications
(interactive and batch)

MATLAB Distributed Computing Engine

Scheduler

Result

Task

Lab

Job

Result

Result

Result

Result

Client Machine

MATLAB

SIMULINK

Toolboxes

Blocksets

Distributed Computing Toolbox

Compute cluster
Large Memory Requirements
Transposing a Distributed Matrix

Using FORTRAN and MPI

Using MATLAB and MPI

Using Distributed Arrays

Transposing a Distributed Matrix

Using FORTRAN and MPI

Subroutine final_assembly(d.ml, ml_comm, master, tag, comm, p, q)

End subroutine final_assembly

Using MATLAB and MPI

transposed = transpose(locals_t);

Preallocate local part of the transposed matrix

T = zeros(size(locals_t));

This part remains on the same lab for all the labs

exchange = locs(k, labindex - 1 : labindex + 1 : labindex + locals_t) + 1

This part generates the sequence in which labs exchange data

labToFrom(labToFrom(labToFrom(labToFrom(labToFrom))));

end

using Distributed Arrays

function Y, locals = transpose(locals, cols);

function Y, locals = transpose(locals, cols);

...
Example:

Image Formation Algorithms: Synthetic Aperture Radar (SAR)

- **Description**
  - SAR is a sophisticated method of post-processing radar data
  - Size and processing requirements demand lots of memory

- **Approach**
  - Processing SAR images involves interdependent 2-D operations
  - Distribute image across the cluster
function ImageOut = cztproc_single

% Read in SAR image data
load sarimage.mat;
[I,N] = size(fftImage);

im_dist = distribute(fftImage,1);

tic;

% azimuth processing

nfft = power(2, nextpow2(2*N-1));
w = exp(-j*2*pi/N);
kx = (-N+1):N-1 ;
kx2 = (kx .^ 2) ./ 2;
ww = w .^ (kx2);  \% \textless;----- Chirp filter is 1./ww
nn = (0:(N-1))';
aa = ww(N+nn);

for i = 1:I,
  \% Perform azimuth CZT
  x = im_dist(i,:);
y = ( x.' ) .* aa;

  fy = fft( y, nfft );
  fv = fft( 1 ./ ww(1:(2*N-1)), nfft );  \% \textless;----- Chirp filter.
y = fy .* fv;
g = ifft( fy );

  g = [g( N:(2*N-1), : ) .* ww( N:(2*N-1) )].'; \% ok<NBRAK>

  im_dist(i,:) = g;
end
function ImageOut = cztproc_single

% Read in SAR image data
load sarimage.mat;
[I,N] = size(fftImage);
im_dist = distribute(fftImage,1);
tic;

% Azimuth processing
nfft = power(2, nextpow2(2*N-1));
w = exp(-j*2*pi/N);
kk = ( (-N+1):N-1 ) .';
kk2 = (kk .^ 2) ./ 2;
ww = w .^ (kk2);  \textbf{\%} <------ Chirp filter is 1./\textit{ww}
nn = (0:(N-1))';

aa = ww(N+nn);

parfor i = 1:I,
  \textbf{\%} Perform azimuth C2T
  x = im_dist(i,:);
y = ( x.' ) .* aa;
  fy = fft( y, nfft );
  fv = fft( 1 ./ ww(1:(2*N-1)), nfft );  \textbf{\%} <------ Chirp filter.
  fy = fy .* fv;
  g = ifft( fy );
  g = [g( N:(2*N-1), : ) .* ww( N:(2*N-1) )].';  \#ok<NBRAK>
im_dist(i,:) = g;
end
Distributed Computing in the Engineering Workflow

Distributed Computing Toolbox

Scheduler

Compute cluster
MATLAB Distributed Computing Engine

Task
Result
Job
Worker

MathWorks
Aerospace and Defense Conference ’07
Why is this important?

- Technical computing
  - Modeling and analysis involves numerous runs
    - Monte carlo or similar applications very common
    - Complexity of algorithms causes longer execution times
  - Data sets are increasing in size

- Model-based design
  - Simulation is done prior to real-world implementation
    - Many scenarios tested
    - Optimal solutions can be found earlier
  - Simulations are growing in complexity and size
    - Simulation time increases
    - Rsim and other targets are only part of the solution
Summary

- Hardware trends are impacting everybody

- Understanding and creating distributed applications will be an important skill for anybody in the fields of Computer Science or Technical Computing

- MathWorks provides a market-leading solution for distributed applications

- Demo available in exhibit hall

MathWorks
Aerospace and Defense Conference ’07