Parallel Computing with MATLAB®

Elwin Chan
## Solving Big Technical Problems

<table>
<thead>
<tr>
<th>Difficulties</th>
<th>You could…</th>
<th>Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long running</td>
<td>Wait</td>
<td>Run similar <em>tasks</em> on independent processors in <em>parallel</em></td>
</tr>
<tr>
<td>Computationally intensive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large data set</td>
<td>Reduce size of problem</td>
<td>Load <em>data</em> onto multiple machines that work together in <em>parallel</em></td>
</tr>
</tbody>
</table>
Parallel Computing

**Difficulties**

- Jobs run in scheduled mode
- Hard to debug
- Cannot access intermediate answers
- Hard to diagnose bottlenecks in algorithm

**Solution**

- Work *interactively* in parallel
Parallel Computing with MATLAB

Pool of MATLAB Workers
Parallel Computing with MATLAB

No code changes
- Implicit Multithreaded MATLAB
- Toolbox Support:
  - Optimization Toolbox™
  - Genetic Algorithm and Direct Search Toolbox™
  - SystemTest™

Trivial changes
- `parfor`
- `job` and `tasks`

Extensive changes
- MATLAB and MPI

Task Parallel

Data Parallel
Agenda

- Speed up algorithms without code changes
  - Develop parallel code interactively
    - Task parallel applications for faster processing
    - Data parallel applications for handling large data sets
  - Schedule your programs to run
  - Tips on developing parallel code
Parallel Computing with MATLAB

No code changes
- Implicit Multithreaded MATLAB
- Toolbox Support:
  - Optimization Toolbox
  - Genetic Algorithm and Direct Search Toolbox
  - SystemTest

Trivial changes
- `parfor`
- `job` and `tasks`

Extensive changes
- MATLAB and MPI

Task Parallel

Data Parallel
Demo: Speed Up Mathematical Operations

```matlab
>> r = rand(1000,1000);
>> % Single-threaded
>> tic; t = r*r; toc
Elapsed time is 1.389135 seconds.
>> % Multi-threaded
>> tic; t = r*r; toc
Elapsed time is 0.728652 seconds.
>>
```
Demo: Speed Up for Implicit Multithreaded Computations

- No change required for user code
- Enables multithreading for key mathematical routines
  - Linear algebra operations
  - Element-wise operations
Implicit Multithreaded Computation

- Linear algebra operations
  - Uses multithreaded Basic Linear Algebra Subroutines (BLAS)
  - BLAS are vendor specific
  - Optimized for specific processor

- Element-wise operations
  - Just-in-time acceleration (JIT) generates on-the-fly multithreaded code
Parallel Computing with MATLAB

No code changes
- Implicit Multithreaded MATLAB
- Toolbox Support:
  - Optimization Toolbox
  - Genetic Algorithm and Direct Search Toolbox
  - SystemTest

Trivial changes
- `parfor`
- `job` and `tasks`

Task Parallel
- Data Parallel
- `distributed`

Extensive changes
- MATLAB and MPI

MathWorks
Aerospace and Defence Conference '08
Demo: Support in Optimization Toolbox

%% Start MATLAB® Pool of labs
%% Requires Parallel Computing Toolbox™
matlabpool open

%% Specify options for parallel optimization
options = optimset('UseParallel','always');

%% Call the optimization routine
x = fmincon(@objfun,x0,[],[],[],[],[],[],LB,UB,@confun,options);
Parallel Support in Optimization Toolbox

- **Functions:**
  - `fmincon`
    finds a constrained minimum of a function of several variables
  - `fminimax`
    finds a minimax solution of a function of several variables
  - `fgoalattain`
    solves the multiobjective goal attainment optimization problem

- Functions can take finite differences in parallel in order to speed the estimation of gradients
SystemTest Supports Parallel Computing for MATLAB and Simulink Applications

Distribute MATLAB and Simulink models for execution on a computer cluster or a multiprocessor system

- Run multiple simulations faster
- Use a checkbox to distribute – no additional code required
- Use homogeneous or heterogeneous platforms
Agenda

- Speed up algorithms without code changes
- Develop parallel code interactively
  - Task parallel applications for faster processing
  - Data parallel applications for handling large data sets
- Schedule your programs to run
Parallel Computing with MATLAB

- Implicit Multithreaded MATLAB
- Toolbox Support:
  - Optimization Toolbox
  - Genetic Algorithm and Direct Search Toolbox
  - SystemTest

Task Parallel
- `parfor`
- `job` and `tasks`

Data Parallel
- `distributed`
- MATLAB and MPI

No code changes
- No code changes

Trivial changes
- Trivial changes

Extensive changes
- Extensive changes

MathWorks
Aerospace and Defence Conference '08
Distributing Tasks (Task Parallel)
Demo: Monte Carlo Simulation of Coin Tossing

10 Simulations of Flipping 20 Coins at a Time

Number of Heads Out of 20
Demo: Monte Carlo Simulation of Coin Tossing

10 Simulations of Flipping 20 Coins at a Time

Number of Heads Out of 20

Change in Number of Heads
Parallel for-Loops

\[
\text{parfor } i = 1 : n \\
\% \text{ do something with } i \\
\text{end}
\]

- Mix task-parallel and serial code in the same function
- Run loops on a pool of MATLAB resources
- Iterations must be order-independent
- M-Lint analysis helps in converting existing for-loops into parfor-loops
Agenda

- Speed up algorithms without code changes
- Develop parallel code interactively
  - Task parallel applications for faster processing
  - Data parallel applications for handling large data sets
- Schedule your programs to run
- Tips on developing parallel code
Parallel Computing with MATLAB

No code changes
- Implicit Multithreaded MATLAB
- Toolbox Support:
  - Optimization Toolbox
  - Genetic Algorithm and Direct Search Toolbox
  - SystemTest

Trivial changes
- `parfor`
- `job` and `tasks`

Extensive changes
- MATLAB and MPI

Task Parallel

Data Parallel
- `distributed`
Large Data Sets (Data Parallel)

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>26</td>
<td>41</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>13</td>
<td>28</td>
<td>43</td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>44</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td>16</td>
<td>31</td>
<td>46</td>
</tr>
<tr>
<td>17</td>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>18</td>
<td>33</td>
<td>48</td>
</tr>
<tr>
<td>19</td>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>20</td>
<td>35</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>36</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>37</td>
<td>52</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>26</td>
<td>41</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>42</td>
</tr>
<tr>
<td>13</td>
<td>28</td>
<td>43</td>
</tr>
<tr>
<td>14</td>
<td>29</td>
<td>44</td>
</tr>
<tr>
<td>15</td>
<td>30</td>
<td>45</td>
</tr>
<tr>
<td>16</td>
<td>31</td>
<td>46</td>
</tr>
<tr>
<td>17</td>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>18</td>
<td>33</td>
<td>48</td>
</tr>
<tr>
<td>19</td>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>20</td>
<td>35</td>
<td>50</td>
</tr>
<tr>
<td>21</td>
<td>36</td>
<td>51</td>
</tr>
<tr>
<td>22</td>
<td>37</td>
<td>52</td>
</tr>
</tbody>
</table>
Demo: Interactive Face Recognition

- Do we recognize this person?
- Compare this image against a database.
- Images in database are represented using six principal eigenfaces (component images).
- Image set must be handled in one bite.
Dataset of Faces

- Single snapshot used to build eigenfaces
- Data set also contains same individuals pulling different expressions – used to test recognition algorithm
- 40 individuals in 10 poses in this dataset

[Face Data provided courtesy of AT&T Laboratories Cambridge]
Face Recognition Algorithm

- Sample faces processed into eigenface components

Compute Mean Face

Subtract

Compute Eigenvectors (Eigenfaces)

Select 6 Principal Eigenfaces

Facial Signatures

\[ Face = \sum_{i=1}^{6} a_i e_i \]
Face Recognition Algorithm

- Sample faces processed into eigenface components

Mean Face

Select 6 Principal Eigenfaces

Identification requires only this Reduced Dataset!
Face Recognition Algorithm

- Test image broken into eigenface components and compared with existing database

- Subtract Test Image
- Find Closest Match (LMS)
- Index
- Reconstruct Match
- Reconstruct Test
- Metric

6 Principal Eigenfaces

Mean Face

Facial Signatures
Distributed Arrays and Parallel Algorithms

- Distributed arrays
  - Store segments of data across participating workers
  - Create from any built-in class in MATLAB
    - Examples: doubles, sparse, logicals, cell arrays, and arrays of structs

- Parallel algorithms for distributed arrays
  - Matrix manipulation operations
    - Examples: indexing, data type conversion, and transpose
  - Parallel linear algebra functions, such as \texttt{svd} and \texttt{lu}
  - Data distribution
    - Automatic, specify your own, or change at any time
MPI-Based Functions in Parallel Computing Toolbox™

Use when a high degree of control over parallel algorithm is required

- High-level abstractions of MPI functions
  - `labSendReceive`, `labBroadcast`, and others
  - Send, receive, and broadcast any data type in MATLAB

- Automatic bookkeeping
  - Setup: communication, ranks, etc.
  - Error detection: deadlocks and miscommunications

- Pluggable
  - Use any MPI implementation that is *binary*-compatible with MPICH2
Agenda

- Speed up algorithms without code changes
- Develop parallel code interactively
  - Task parallel applications for faster processing
  - Data parallel applications for handling large data sets
- Schedule your programs to run
  - Tips on developing parallel code
Distributed Applications

MathWorks
Aerospace and Defence Conference '08
Demo: Scheduled Monte Carlo Coin

\[
\text{>> createJob(...)}
\]
\[
\text{>> createTask(...)}
\]
Demo: Scheduled Monte Carlo Coin using parfor

```matlab
>> createMatlabPoolJob
```
Demo: Scheduled Face Recognition

`>> createParallelJob`
Options for Scheduling Jobs

**Task Parallel**

```matlab
>> createMatlabPoolJob
or
>> batch
```  

**Data Parallel**

```matlab
>> createParallelJob
>> createJob(…)
>> createTask(…)
```
Dependencies

- **job – FileDependencies**
  - Files are copied from client to each worker machine
  - Zip compressed
  - Uncompressed and added to the MATLAB path
  - Convenient for .m files, but can be slow for large data files

- **job – PathDependencies**
  - Shared directories are added to the MATLAB path
  - Mixing of Windows® and UNIX® paths allowed
  - Reduces the amount of data transfer from client to cluster
Configurations

- Save environment-specific parameters for your cluster

Benefits
- Enter cluster information only once
- Modify configurations without changing MATLAB code
- Apply multiple configurations when running within same session
Run *Four Local* Workers with a Parallel Computing Toolbox License

- Easily experiment with explicit parallelism on multicore machines
- Rapidly develop parallel applications on local computer
- Take full advantage of desktop power
- Separate computer cluster not required
Scale Up to Cluster Configuration with No Code Changes

Parallel Computing Toolbox

Compu

ter Cluster

MATLAB Distributed Computing Server

Scheduler

Worker

Worker

Worker

MathWorks
Aerospace and Defence Conference '08
Agenda

- Speed up algorithms without code changes

- Develop parallel code interactively
  - Task parallel applications for faster processing
  - Data parallel applications for handling large data sets

- Schedule your programs to run

Tips on developing parallel code
Development and Debugging Process

Develop serial code normally on local machine

Task or Data Parallel?

Task Parallel

parfor / Jobs and Tasks

Run on local workers

Data Parallel

pmode / Distributed Arrays

Run on N cluster nodes
Parallel Profiler

- Profiles the execution time for a function
  - Similar to the MATLAB profiler
  - Includes information about the communication between labs
    - Time spent in communication
    - Amount of data passed between labs

- Benefits
  - Identify the bottlenecks in your parallel algorithm
  - Understand which operations require communication
Factors to Consider for Speeding Up Your Code

- Share code and data with workers efficiently using `FileDependencies` or `PathDependencies`

- There is always an overhead to distribution
  - Don’t make a task too small
  - Combine small repetitive function calls into one larger one

- Use the M-lint and parallel profiler (`mpiprofile`) to identify slow code

- Minimize I/O
Summary

- Speed up algorithms without code changes
- Develop parallel code interactively
  - Task-parallel applications for faster processing
  - Data-parallel applications for handling large data sets
- Schedule your programs to run
- Tips on developing parallel code