Matlab® and MUGrid
From Compilation to Distribution

David J. Herzfeld
david.herzfeld@marquette.edu

Marquette University
Department of Biomedical Engineering

July 15, 2010
Why Matlab®?

- Matlab® is used extensively at MU.
  - Computational & Financial Modeling
  - Rapid Prototyping
  - Comparatively “Easy” to Learn
- Matlab® problems can quickly become large (parameter exploration).
- Ideal candidate for high throughput computing!
Outline

1. Introduction

2. Matlab Compiler

3. Matlab & Condor
We won’t be covering...

- In-depth Matlab® Programming
- Matlab® Parallel Computing Toolbox
- Matlab® Distributed Computing Server
Assumed Knowledge

- Basic understanding of shell scripting (Bash)
  - See notes from Sessions 1 & 2
- Basic understanding of Condor® HTC
  - See notes from Session 3
  - [http://www.cs.wisc.edu/condor](http://www.cs.wisc.edu/condor)
- Basic understanding of Matlab®
Environment & Examples

- Examples will be using Père
  - Techniques can be generalized to the rest of MUGrid and beyond
- Login to Père
  - From Linux/Unix/Mac: `ssh username@pere.marquette.edu`
  - Use an SSH client from Windows

Copy Examples

```bash
$ cp -r /group/hpc-bootcamp/2010Summer/Session5 ~/matlab_examples
```
“Hello World!” for Matlab®

disp(['Hello World!']);
disp(['My random number: ' num2str(rand())]);
Where’s Matlab®?

$ which matlab
/usr/bin/which: no matlab in \\
(opt/pgi/linux86-64/10.3/bin:
/usr/mpi/gcc/openmpi-1.2.8/bin:
/usr/kerberos/bin:
/opt/condor-7.4.1/bin:
/opt/condor-7.4.1/sbin:
/usr/local/bin:/bin:/usr/bin:
/home/MARQNET/8589herzfed/bin)
Modules on Père

- Use modules to set paths and environment variables
- See http://modules.sourceforge.net
- List available modules: module avail
- Notice the matlab module
example1a.m on Père (hpc-cn1)

Running example1a.m

$ module load matlab
$ which matlab
/group/hpc-share/matlab/bin/matlab
$ matlab -nodesktop -nosplash -nojvm -nodosplay
>> example1
Hello World!
My random number: 0.81472
example1a.m Required Steps

1. Set up the environment
2. Start Matlab®
3. Run the script
General Required Steps

1. Check for required input files & code
2. Set up the environment
3. Start Matlab®
4. Run the script
5. Tear down:
   1. Stop Matlab®
   2. Remove temporary/unneeded files
Creating a Bash Script

example1a.sh

#!/bin/bash
if [ ! -f example1a.m ]
then
    echo "ERROR: Unable to find example1a.m" >&2
    exit 1
fi

module load matlab
matlab -nodesktop -nojvm -nodisplay -r example1a
# Note: script exits with status of the matlab command
# Note: We do not want the .m extension here
Running `example1a.sh`

**Shell Commands**

```
$ chmod +x example1a.sh
$ ./example1a.sh
```
Issues

- Matlab® did not close after showing the random number
- If we could not see the output, it would look like the program hung
- Potential solutions:
  - Add an exit at the end of the script
  - Pipe the contents to stdin

```bash
$ matlab -nodesktop -nojvm -nodisplay < example1a.m
```
example1a.m → example1b.m

disp(['Hello World!']);
disp(['My random number: ' num2str(rand())]);
exit;
Current Problem

- *example1a.m* and *example1b.m* require a working copy of Matlab®
- Concurrent use of Matlab® or toolbox licenses may be an issue on some systems
- Matlab® is “expensive”
- Due to the tightly-coupled nature of current clusters, Matlab® Distributed Computing Server would be required = $$$
Potential Solutions

- Simple Matlab® Scripts:
  - Try a Matlab® alternative
    - octave
    - ANSI C
    - Python with NumPy & SciPy

- Advanced Matlab® Scripts:
  - Matlab® Compiled Executable
Introduction to the Matlab® Compiler

- Matlab® Compiler 4.13
- Limited GUI Support
Guidelines for Deployable Code

- Dynamic code is **NOT** allowed
  - **NO** calling of specific files via modified paths
  - **NO** help
  - **NO** loadlibrary unless you provide function prototypes

- Common pitfalls:
  - Mismatched Matlab® runtime (MCR) and development environments
  - Global variables whose names are used in other functions
Another Simple Example

example2.m

function m = example2(n)
% Returns a random matrix of size n
if ischar(n)
    n = str2num(n);
end
m = rand(n)
Makefile

example2: example2.m
    mcc -m -R -nodisplay -R -nojvm example2.m

clean:
    rm -f example2
    rm -f *_main.c *_data.c *.prj readme.txt *.log run_*.sh
mcc Options

- **m**: Generate a standalone executable (not a library)
- **R**: Specify runtime options
  - **nodisplay**: No need to display anything.
  - **nojvm**: Since there is no display, there is little need for Java

Additional Useful Flags:
- **a**: Add a filename or folder to an archive
- **I**: Include a path when searching for m files (include external toolboxes, for example)
- **l**: Use instead of **m** to generate a library rather than a standalone executable

See **mcc -help** for more details
Running example2

On Père (hpc-hn1)

```
$ make
$ ./example2 5
```

On Other Nodes

```
$ module load matlab
$ make
$ make
$ ./example2 5
```
Bash Script for Compiled Matlab® Executable

```bash
#!/bin/bash
if [ ! -f example2 ]
then
    echo "ERROR: Unable to find example2" >&2
    exit 1
fi
if [ ! -x example2 ]
then
    chmod +x example2
fi
module load matlab
./example2 $1
```
When to Distribute Matlab® Code?

- Statistical analysis of simulation/model outcome ("I need to run this Matlab® model 500 times")
- Problem is composed of independent tasks (parameter exploration)
- Datasets are very large (memory limitations), but some/all analyses can be performed independently
- Many others...
Why Condor® & Matlab®?

- Independent tasks are ideal for high throughput computing (HTC).
- Condor® is a great HTC scheduler
- Matlab® + Condor® = ❤
- Identical Condor® techniques can be used across MUGrid
- Scales to national grids for really big problems.
  - Topic of Session 7!
\[ X = \sum_{i=1}^{N} \sqrt{A_i B_i + C} \]

- \( A_i, B_i, \) and \( C \) are square.
- \( i \) is very large (10,000+)
- Loading a .mat file containing all \( A \) and \( B \) gives:

```
??? Error using ==> load
Out of memory. Type HELP MEMORY for your options.
```
Serial Matlab® Code

example3.m

```matlab
function [] = example3()
load('A.mat'); % Get A (3D matrix)
load('B.mat'); % Get B (3D matrix)
load('C.mat'); % Get C (matrix)
M = size(A, 1);
X = zeros(M); % Create MxM matrix
for i = 1:size(A, 3)
    X = X + sqrt(A(:,:,i) * B(:,:,i) + C);
end
disp(X);
save('X.mat', 'X');
exit;
```

David J. Herzfeld (Marquette University)
Distributing example 3

- The calculation of each $X_i$ are independent
- We only need all of the $X_i$'s during a final summation step

**Distribution Basics:**

- Distributing for and while loops are easy ways of increasing throughput! Especially if -

```matlab
my_file.m

for i = 1:N
    ...
    % LOTS OF COMPUTATIONAL WORK
    ...
end
```
Distributing example3

- Need A, B, and C for each job
  - Distribute $A_i$ and $B_i$ with the job
  - Distribute C to every job
  - Could use .mat or flat text file
  - Matlab® loads $A_i$, $B_i$, and C then computes $X$
Distributing example3

- Performing the summation
  1. Save $X$ for each job and transfer the file back
  2. Matlab® script that runs after distributed work is complete.
  3. Condor®-ized Matlab® job.
    - Stick around for Session 6 to learn how to “set it and forget it”
function [] = example4(i)
if ischar(i)
    i = str2num(i);
end
load(['A_' num2str(i) '.mat']);
load(['B_' num2str(i) '.mat']);
load('C.mat');
X = sqrt(A * B + C);
disp(X);
save(['X_' num2str(i) '.mat'], 'X');
exit;
Condor® Submit File

```bash
example4.sub

Universe = vanilla
Executable = example4.sh
Arguments = $(PROCESS)
Output = example4_$(PROCESS).out
Error = example4_$(PROCESS).err
Log = example4.log
Requirements = ( OpSys == "LINUX" && Arch == "X86_64" )
transfer_input_files = example4, \
    A_$(PROCESS).mat, B_$(PROCESS).mat, C.mat
should_transfer_files = TRUE
when_to_transfer_output = ON_EXIT
Queue 50
```
Performing the Summation

example5.m

function [] = example5(N)
if ischar(N)
    N = str2num(N);
end
load('X_0.mat', 'X');
M = length(X); % Determine M
x_sum = zeros(M);
for i = 0:N-1
    load(['X_' num2str(i) '.mat'], 'X'); % Load X
    x_sum = x_sum + X;
end
disp(x_sum);
exit;
Performing the Summation

**example5.sub**

Universe = vanilla
Executable = example5.sh
Arguments = 50
Output = example5.out
Error = example5.err
Log = example5.log
Requirements = ( OpSys == "LINUX" && Arch == "X86_64" )
transfer_input_files = example5, X.tar.gz
should_transfer_files = TRUE
when_to_transfer_output = ON_EXIT
Queue 1
Performing the Summation

example5.sh

#!/bin/bash
source /etc/bashrc && source /etc/profile
# Source the modules script
source /cluster/Modules/3.2.7/init/bash
# Do Error Checking
tar xzf X.tar.gz
# More Error Checking
chmod +x example5
module load matlab
./example5 $1
# Cleanup
rm -rf X_*.mat
Concluding Remarks

- Condor® can help solve many Matlab® problems quickly
- The Matlab® compiler (mcc) can be used to distribute code across MUGrid
- Management of Matlab® data is required for distribution
Stay tuned for...

- Session 6: Advanced High-Throughput Computing (1:30 - 2:45)
- Combine multiple Matlab® jobs into a workflow using a DAG