

Parallel Matlab(*)

Dr. Guy Tel-Zur

(*)=and other tools

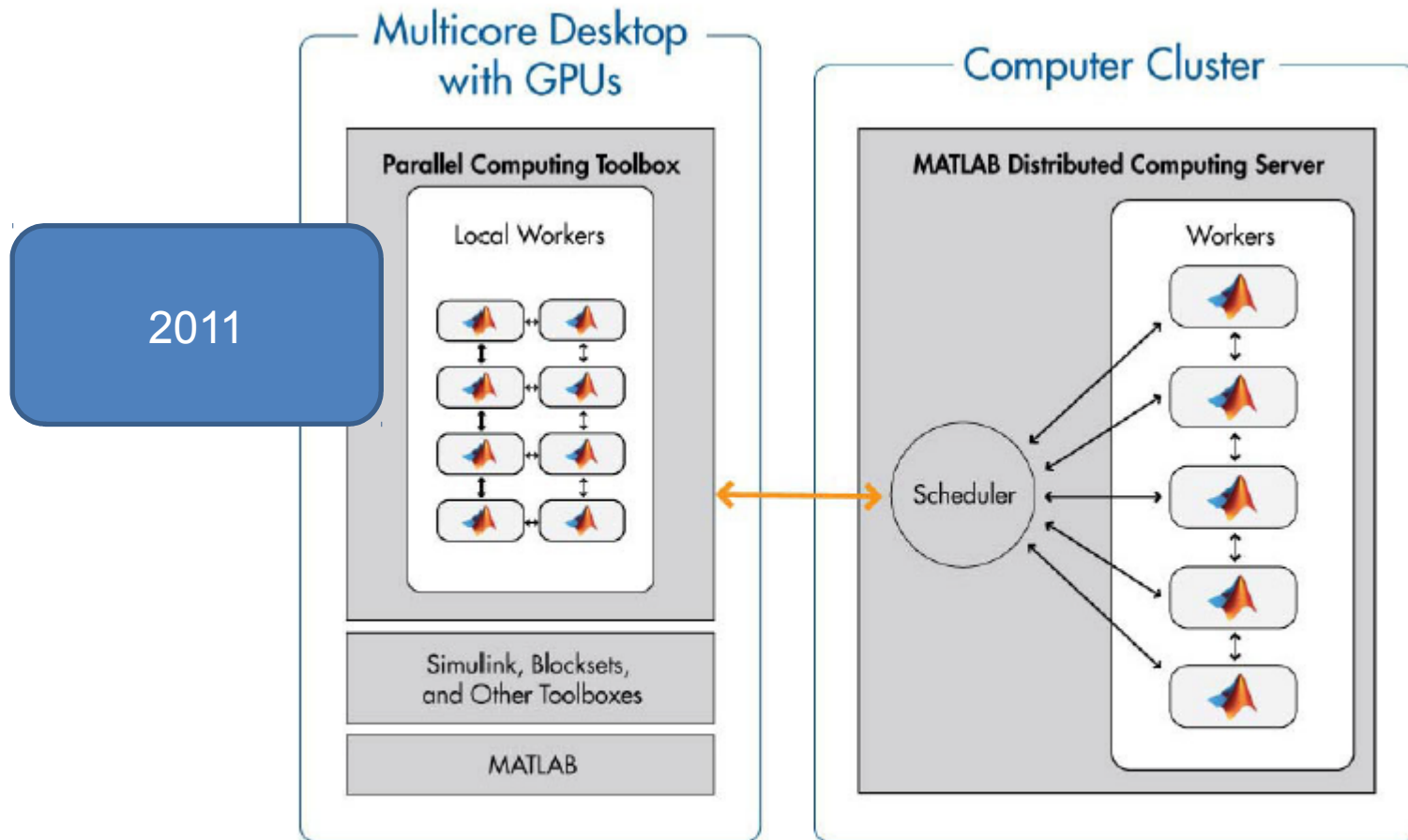
Agenda

- Mathworks – Parallel Computing toolbox
- Parallel Computing with Matlab on Amazon Cloud
- Matlab over GPGPU
- Matlab (Octave) + HTCondor (we will have to learn HTCondor first)
- Parallel Matlab (Octave) using MatlabMPI
- Parallel Matlab (Octave) using pMatlab

Mathworks – Parallel Computing toolbox

- Parallel Computing without CUDA or MPI(...)
- The toolbox provides “workers” (MATLAB computational engines) to execute applications locally on a multicore desktop
- Parallel for-loops (**parfor**) for running task-parallel algorithms on multiple processors
- Computer cluster and grid support (with MATLAB Distributed Computing Server)

Parallel Computing toolbox



Parallel computing with MATLAB. You can use Parallel Computing Toolbox to run applications on a multicore desktop with eight workers available in the toolbox, take advantage of GPUs, and scale up to a cluster (with MATLAB Distributed Computing Server).

Multicore Desktop with GPUs

Parallel Computing Toolbox

12 Local Workers



Simulink, Blocksets,
and Other Toolboxes

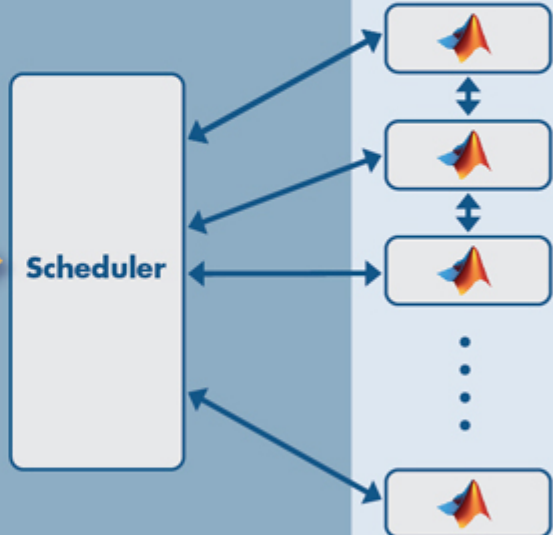
MATLAB®

2012

Computer Cluster

MATLAB Distributed Computing Server

Workers



When installing Matlab check the Parallel Computing Toolbox

Select products to install (recommended products are preselected)

<input type="checkbox"/>	Product	Notes
<input type="checkbox"/>	MATLAB Compiler SDK 6.7	Download Required
<input type="checkbox"/>	MATLAB Report Generator 5.7	Download Required
<input type="checkbox"/>	Mixed-Signal Blockset 1.1	Download Required
<input type="checkbox"/>	Model Predictive Control Toolbox 6.3.1	Download Required
<input type="checkbox"/>	Navigation Toolbox 1.0	Download Required
<input checked="" type="checkbox"/>	Optimization Toolbox 8.4	Download Required
<input checked="" type="checkbox"/>	Parallel Computing Toolbox 7.1	Download Required
<input checked="" type="checkbox"/>	Partial Differential Equation Toolbox 3.3	Download Required
<input type="checkbox"/>	Phased Array System Toolbox 4.2	Download Required
<input type="checkbox"/>	Powertrain Blockset 1.6	Download Required
<input type="checkbox"/>	Predictive Maintenance Toolbox 2.1	Download Required
<input type="checkbox"/>	Reinforcement Learning Toolbox 1.1	Download Required
<input type="checkbox"/>	RF Blockset 7.3	Download Required
<input type="checkbox"/>	RF Toolbox 3.7	Download Required
<input checked="" type="checkbox"/>	Risk Management Toolbox 1.6	Download Required
<input checked="" type="checkbox"/>	Robotics System Toolbox 3.0	Download Required

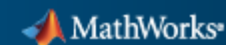


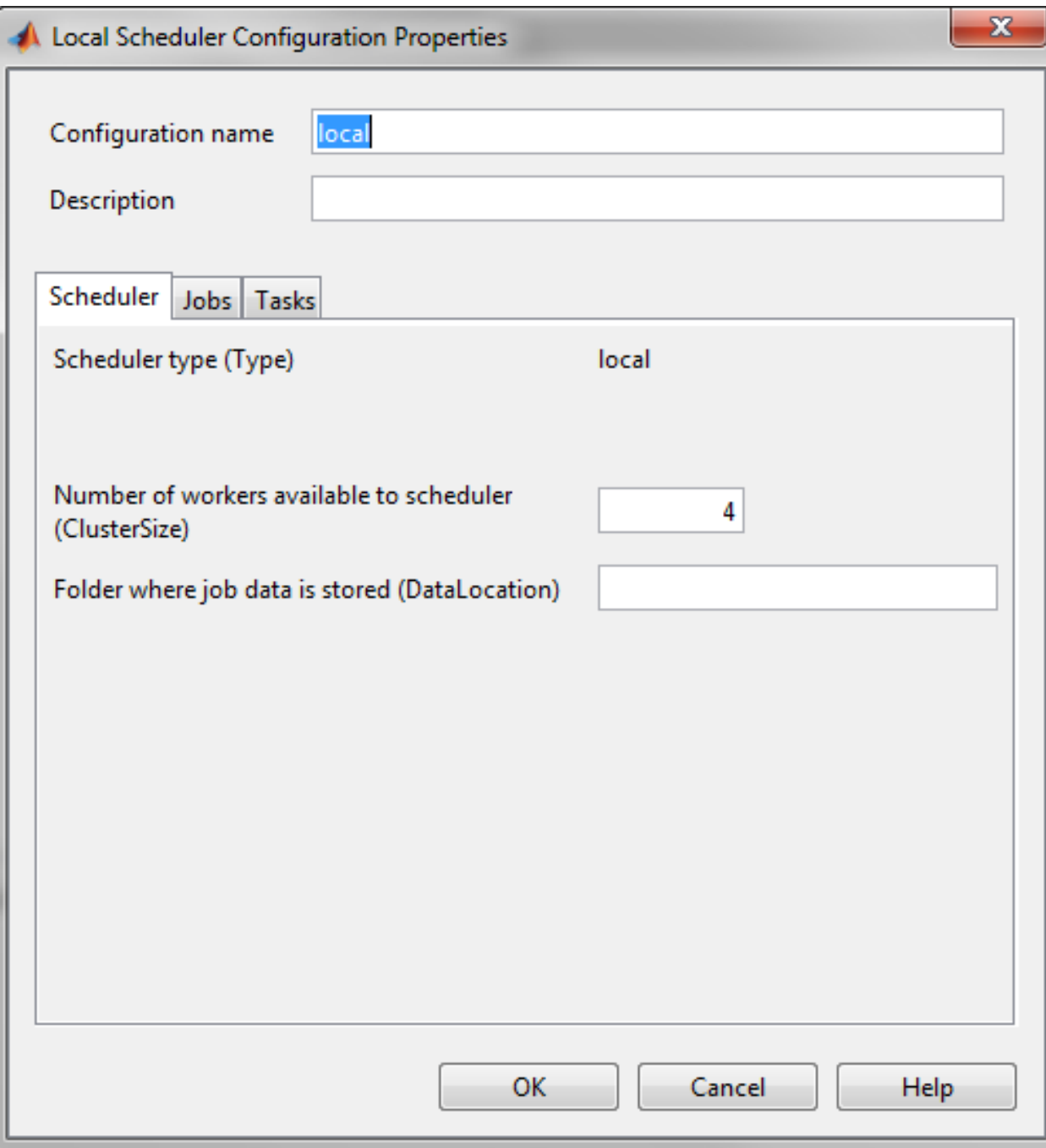
< Back

Next >

Cancel

Help





The image shows a Windows-style dialog box titled "Local Scheduler Configuration Properties". It has a standard title bar with a close button (X) in the top right corner. The dialog is divided into several sections. At the top, there are two text input fields: "Configuration name" with the value "local" and "Description" which is empty. Below these is a tabbed interface with three tabs: "Scheduler" (selected), "Jobs", and "Tasks". The "Scheduler" tab contains a large container with the following elements: a label "Scheduler type (Type)" with the value "local", a label "Number of workers available to scheduler (ClusterSize)" with a text box containing the value "4", and a label "Folder where job data is stored (DataLocation)" with an empty text box. At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

Local Scheduler Configuration Properties

Configuration name

local

Description

Scheduler

Jobs

Tasks

Scheduler type (Type)

local

Number of workers available to scheduler (ClusterSize)

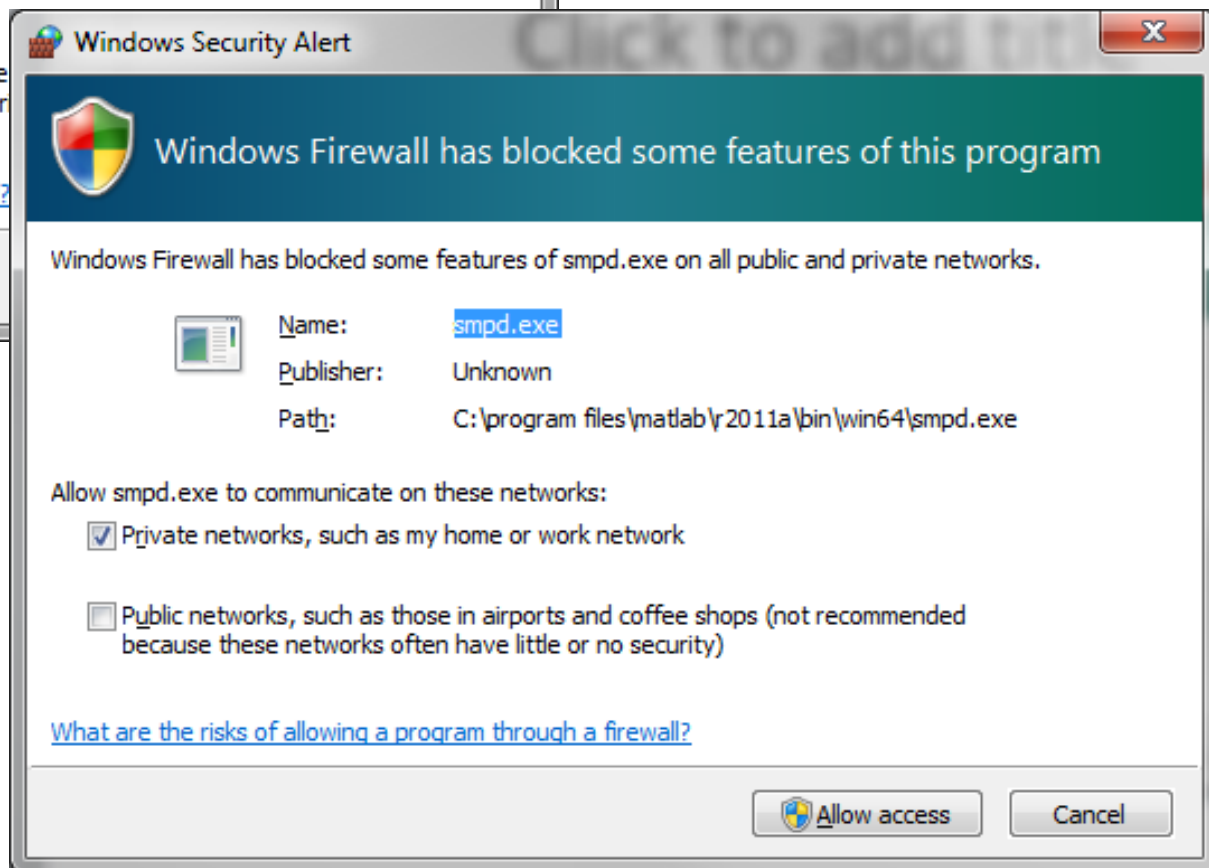
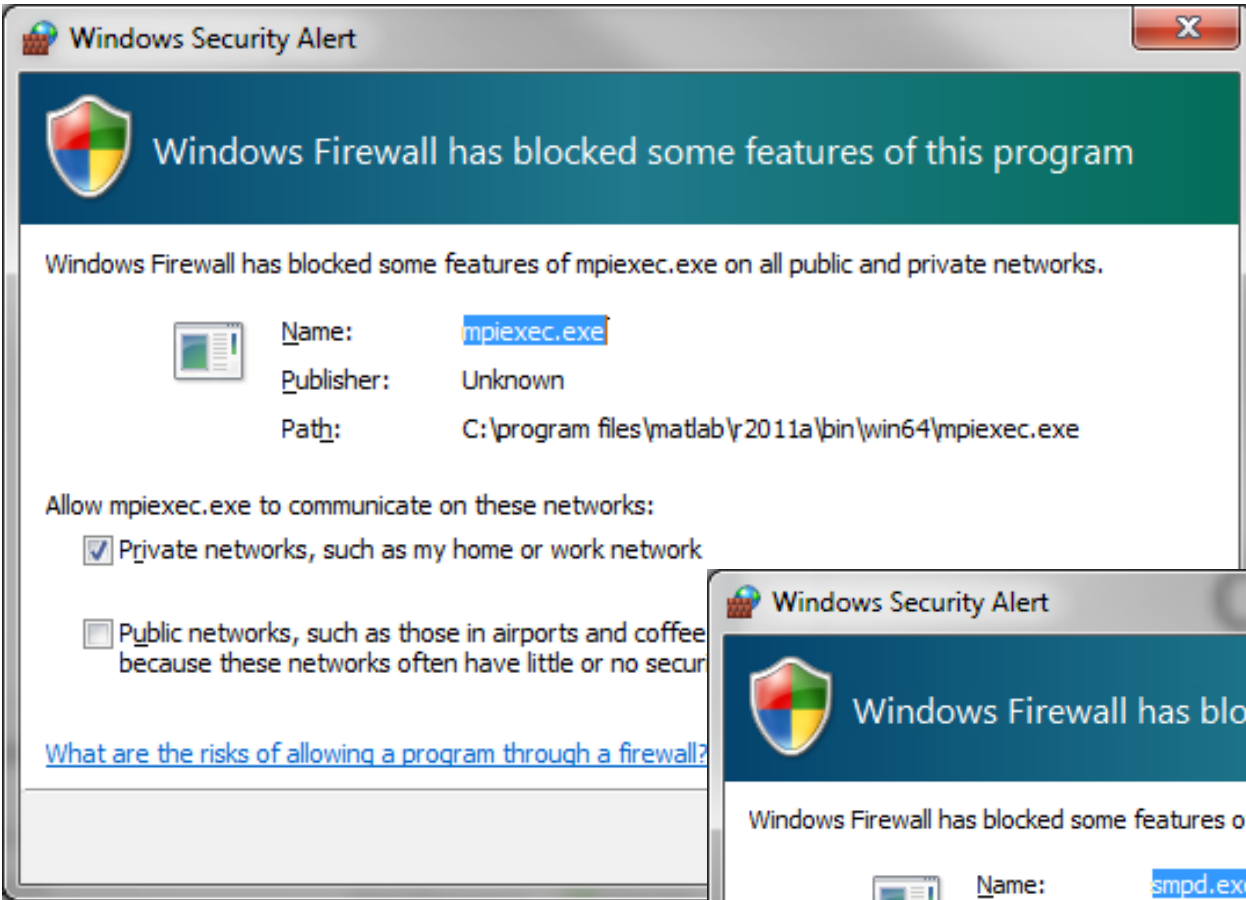
4

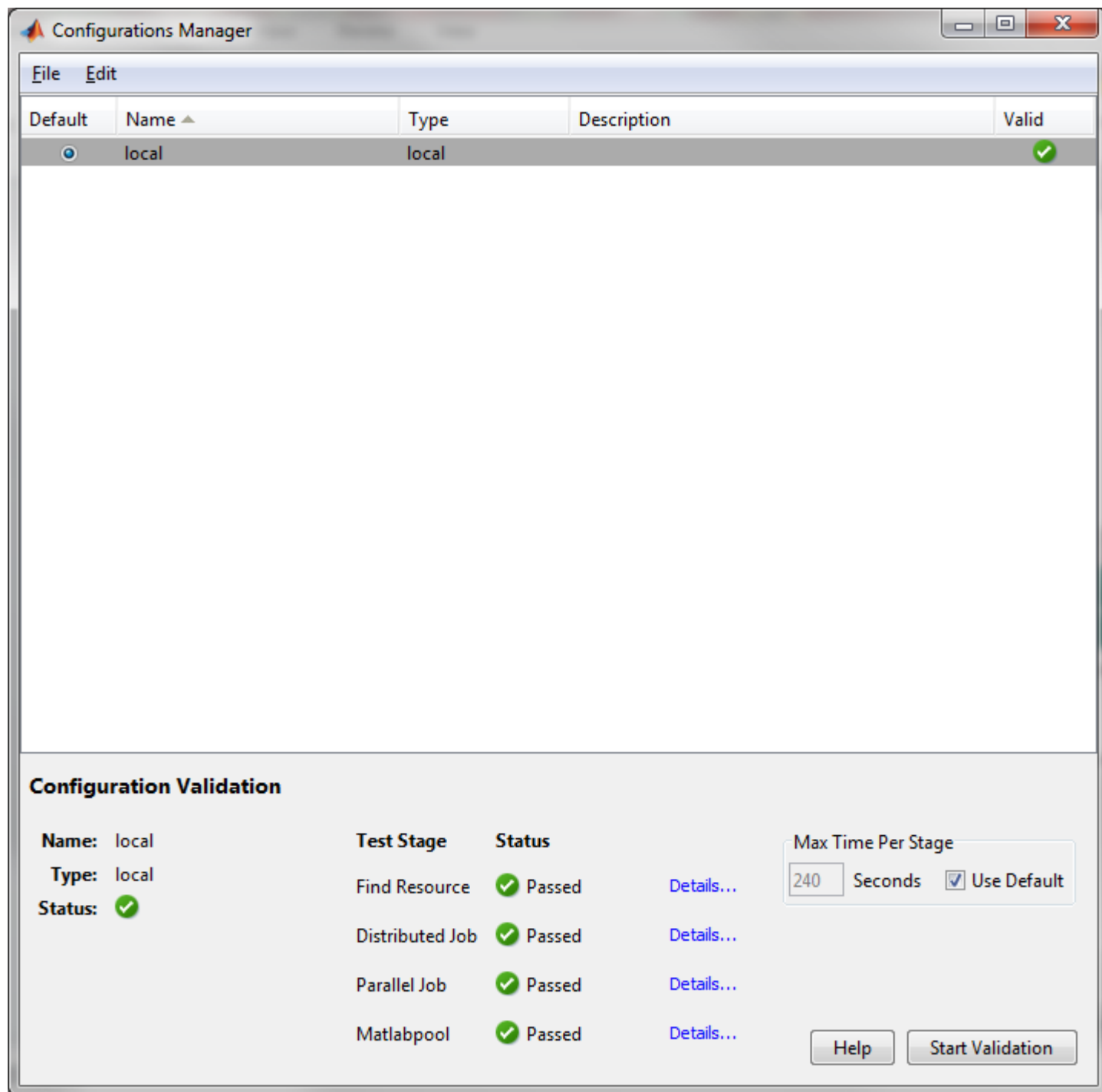
Folder where job data is stored (DataLocation)

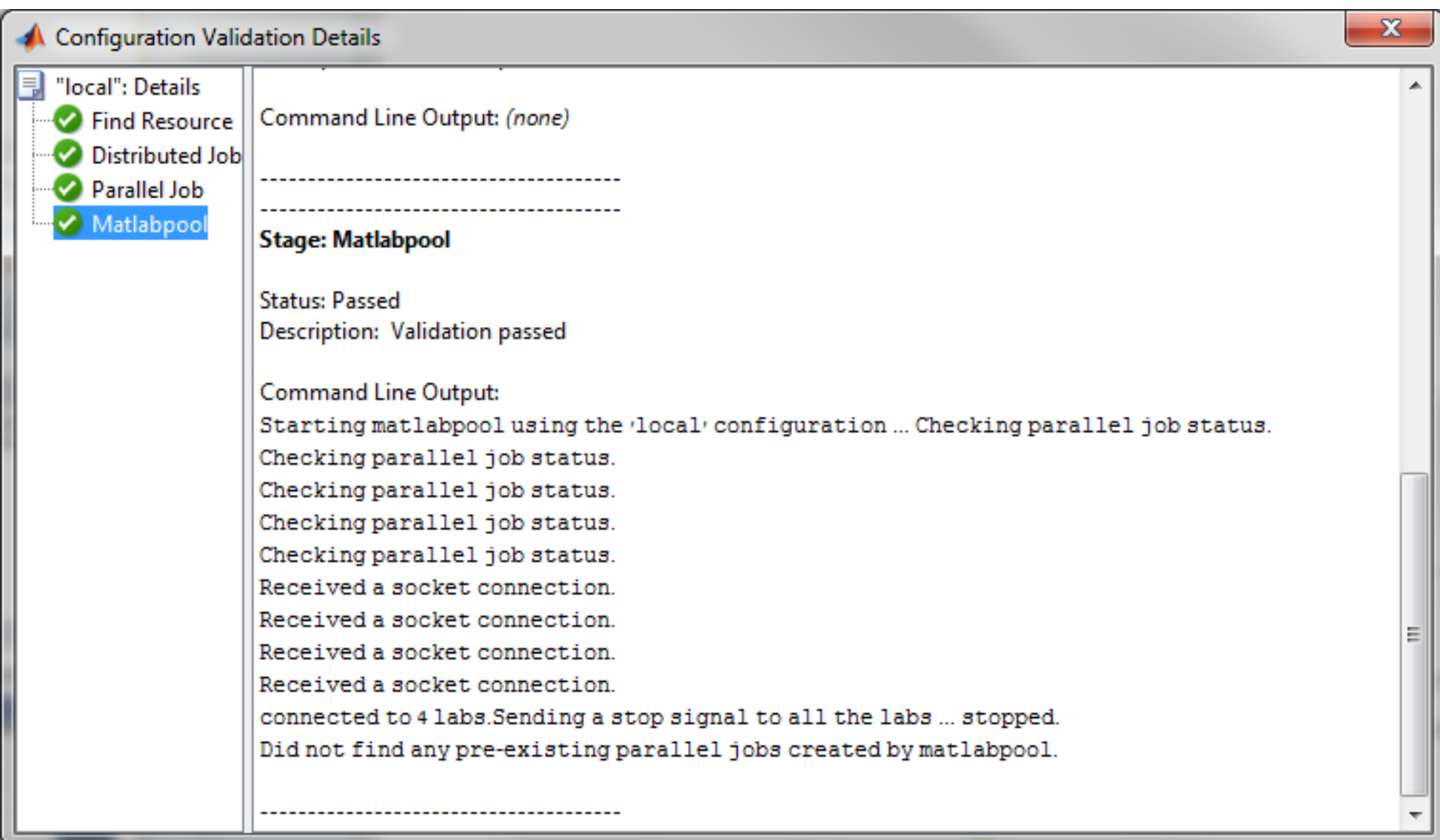
OK

Cancel

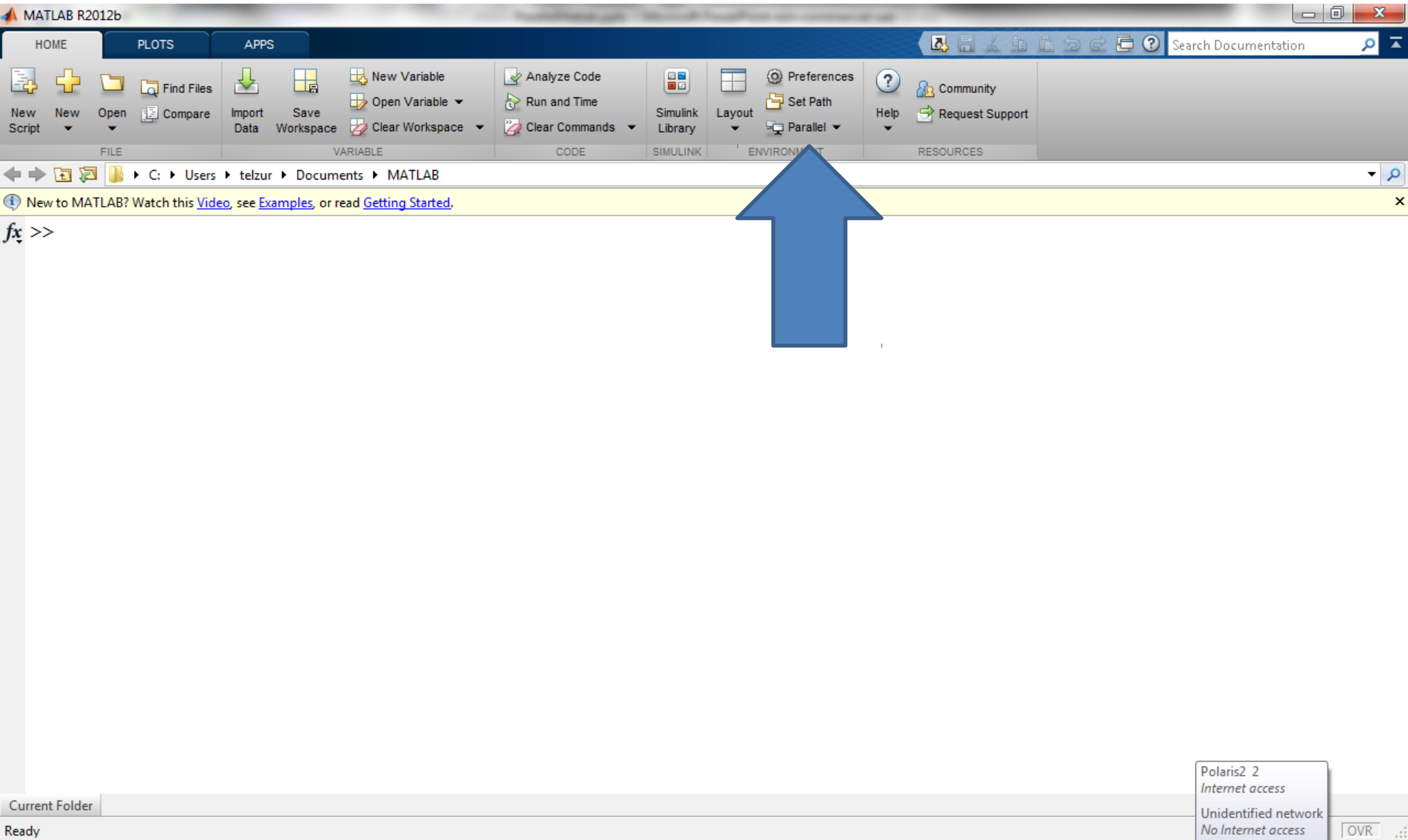
Help











Matlab 2012B



Cluster Profile Manager

Cluster Profile: local (default) Scheduler Type: Local

Overall Status:  Running

Stage	Status	Description
Cluster connection test (parcl...	 Passed	
Job test (createJob)	 Passed	
SPMD job test (createCommu...	 Running	
Pool job test (createCommuni...	--- Not run	
MATLAB pool test (matlabpool)	--- Not run	

Stop Show Details

ZONEALARM

REPEAT PROGRAM

smpd.exe is trying to access the Internet.

SmartDefense Advisor Recommendation:
Advice is not yet available for this program.

☒ Remember this setting

Show More Info

Allow Deny

Windows Task Manager

File Options View Help

Applications Processes Services Performance Networking Users

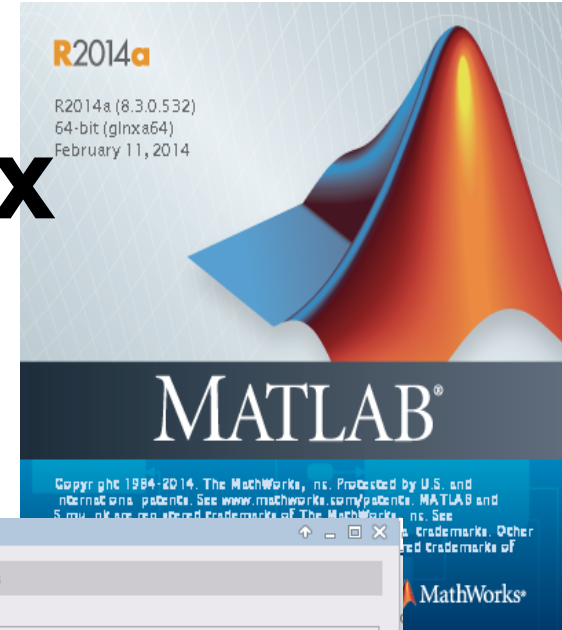
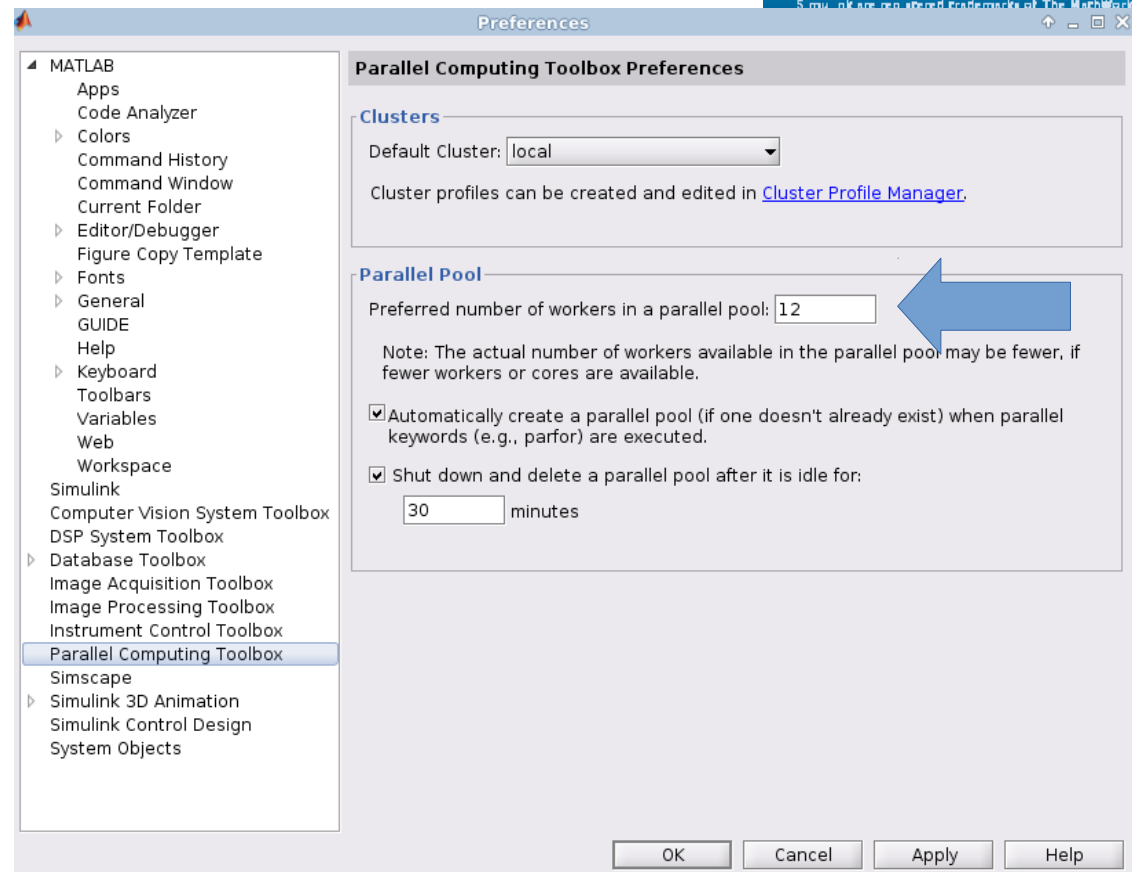
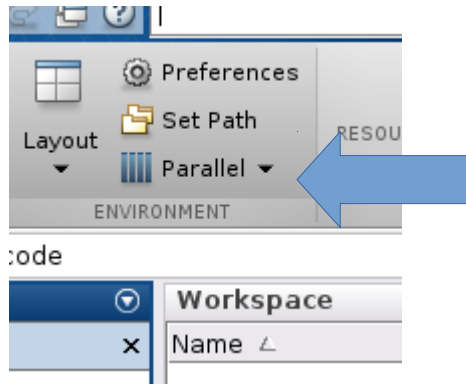
Image Name	User Name	CPU	Memory (P...	Threads	I/O Writes	Image Path Name	Description
System Idle Process	SYSTEM	49	24 K	4			Percentage of time the proc
MATLAB.exe	telzur	21	158,892 K	28	18	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
MATLAB.exe	telzur	20	159,820 K	28	18	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
MATLAB.exe	telzur	03	302,492 K	45	1,453	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
chrome.exe *32	telzur	02	61,620 K	14	32,192	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
svchost.exe	NETWO...	02	8,428 K	28	8,166	C:\Windows\System32\svchost.exe	Host Process for Windows S
taskmgr.exe	telzur	01	3,428 K	8		C:\Windows\System32\taskmgr.exe	Windows Task Manager
POWERPNT.EXE *32	telzur	00	57,004 K	15	1,389		
chrome.exe *32	telzur	00	6,660 K	14	1,737	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
svchost.exe	LOCAL ...	00	444 K	4		C:\Windows\System32\svchost.exe	Host Process for Windows S
mpiexec.exe	telzur	00	3,572 K	2	106	C:\Program Files\MATLAB\R2012b\bin\win64\mpiexec.exe	mpiexec.exe
chrome.exe *32	telzur	00	81,452 K	13	16,359	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
splwow64.exe	telzur	00	3,368 K	7	124	C:\Windows\splwow64.exe	Print driver host for 32bit ap
chrome.exe *32	telzur	00	16,304 K	14	33,421	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
chrome.exe *32	telzur	00	20,916 K	13	18,900	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
TosBtHSP.exe *32	telzur	00	960 K	6		C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosBtHSP.exe	TosBtHSP
chrome.exe *32	telzur	00	39,468 K	13	22,828	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
conhost.exe	telzur	00	1,824 K	1		C:\Windows\System32\conhost.exe	Console Window Host
slimsvc.exe *32	SYSTEM	00	528 K	7	4	C:\Program Files (x86)\CheckPoint\SSL Network Extender\slimsvc.exe	slimsvc.exe
chrome.exe *32	telzur	00	5,688 K	13	446	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
avgcsrva.exe	SYSTEM	00	1,548 K	15	583,957	C:\Program Files (x86)\AVG\AVG2013\avgcsrva.exe	AVG Scanning Core Module
TosBtMng.exe *32	telzur	00	2,088 K	49	167	C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosBtMng.exe	Bluetooth Manager
TosAVRC.exe *32	telzur	00	720 K	3		C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosAVRC.exe	TosAVRC
taskeng.exe	SYSTEM	00	2,240 K	6		C:\Windows\System32\taskeng.exe	Task Scheduler Engine
smpd.exe	telzur	00	3,544 K	6	7	C:\Program Files\MATLAB\R2012b\bin\win64\smpd.exe	smpd.exe
chrome.exe *32	telzur	00	9,484 K	13	6,717	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
smpd.exe	telzur	00	4,080 K	8	3	C:\Program Files\MATLAB\R2012b\bin\win64\smpd.exe	smpd.exe
chrome.exe *32	telzur	00	46,536 K	13	5,582	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
chrome.exe *32	telzur	00	5,408 K	13	105	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
UNS.exe *32	SYSTEM	00	1,064 K	13	2	C:\Program Files (x86)\Intel\Intel(R) Management Engine Components\UNS\UNS.exe	User Notification Service

☒ Show processes from all users

End Process

Processes: 135 CPU Usage: 51% Physical Memory: 78%

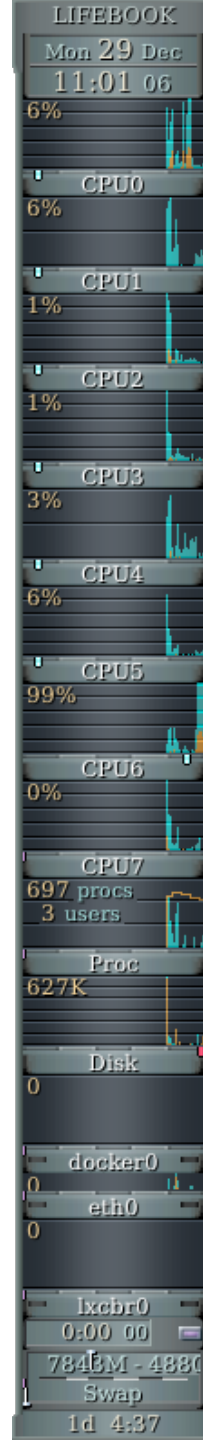
Version 2014a on Linux



Version 2014a on Linux

The screenshot shows the 'Cluster Profile Manager' application window. The title bar reads 'Cluster Profile Manager'. The menu bar includes 'Add', 'Discover Clusters', 'Import', 'Edit', 'Duplicate', 'Delete', 'Rename', 'Set as Default', 'Export', 'Validate', and 'Help'. The main content area is divided into two panes. The left pane, titled 'Cluster Profile', lists 'local (default)'. The right pane, titled 'local', shows the 'Validation Results' tab. It displays an 'Overall Status: Passed' with a green checkmark. Below this is a table with three columns: 'Stage', 'Status', and 'Description'. The table lists five tests, all of which passed. At the bottom right of the right pane are 'Validate' and 'Show Details' buttons.

Stage	Status	Description
Cluster connection test (parcluster)	Passed	
Job test (createJob)	Passed	
SPMD job test (createCommunicatingJob)	Passed	
Pool job test (createCommunicatingJob)	Passed	
Parallel pool test (parpool)	Passed	



parfor - Parallel for loop

parfor - Parallel for loop

Syntax

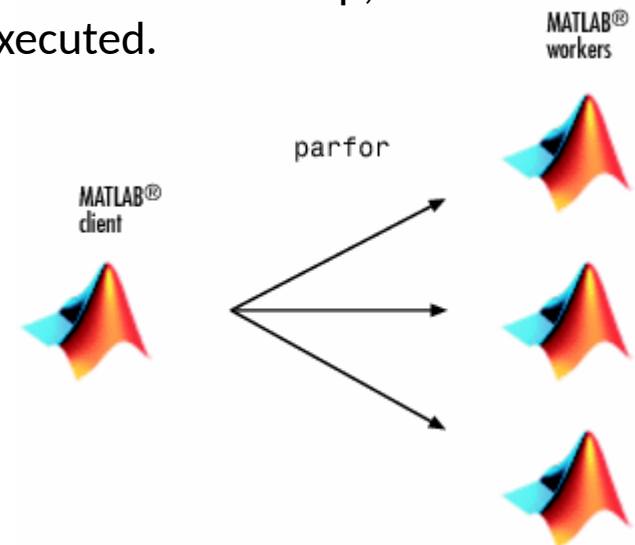
```
parfor loopvar = initval:endval; statements; end  
parfor (loopvar = initval:endval, M); statements; end
```

Description

`parfor loopvar = initval:endval; statements; end` executes a series of MATLAB commands denoted here as *statements* for values of *loopvar* between *initval* and *endval*, inclusive, which specify a vector of increasing integer values. Unlike a traditional for-loop, there is no guarantee of the order in which the loop iterations are executed.

The general format of a `parfor` statement is:

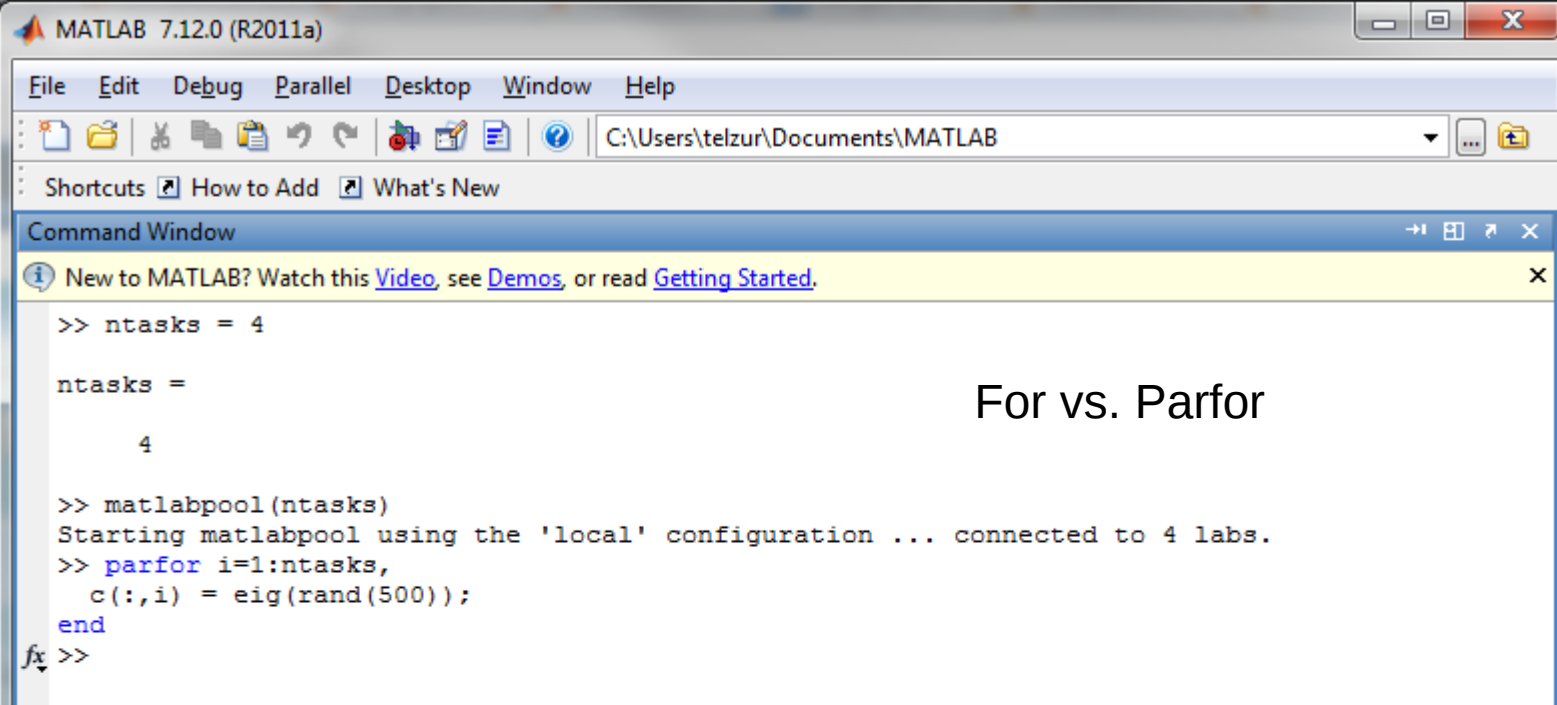
```
parfor loopvar = initval:endval  
    <statements>  
end
```



parfor – an example

Perform three large eigenvalue computations using three computers or cores:

```
ntasks = 4  
matlabpool(ntasks)  
parfor i=1:ntasks,  
    c(:,i) = eig(rand(500));  
end
```



For vs. Parfor

```
>> ntasks = 4;
>> tic;for i=1:ntasks,
c(:,i)=eig(rand(1000));
end; toc
Elapsed time is 18.545340 seconds.
>> tic;parfor i=1:ntasks,
c(:,i)=eig(rand(1000));
end; toc
Elapsed time is 10.980618 seconds.
>>
```

```
Editor - /home/telzur/Documents/Teaching/BGU/PP/PP2015A/lect
parallel1.m x parallel0.m x +
1 - disp('Serial computation');
2 - ntasks=4;
3 - tic; for i=1:ntasks, c(:,i)=eig(rand(1000));
4 - end; toc
5
6 - size(c)
7
8 - disp('parallel computation');
9 - delete(gcp);
10 - parpool('local');
11 - tic; parfor i=1:ntasks, d(:,i)=eig(rand(1000));
12 - end; toc
13 - matlabpool('close')
14
15 - size(d)
```

Demo: .../lecture09/code/parallel0.m

4 tasks

```
>>
>> parallel0
Serial computation
Elapsed time is 3.374473 seconds.
ans =

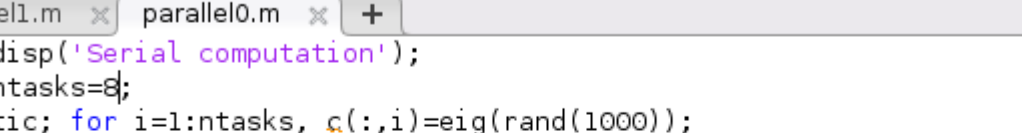
    1000     4

parallel computation
Starting parallel pool (parpool) using the 'local' profile ... co
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ... co
Elapsed time is 2.445419 seconds.
Warning: matlabpool will be removed in a future release.
To shutdown a parallel pool use 'delete(gcp('nocreate'))'
instead.
Parallel pool using the 'local' profile is shutting down.
ans =

    1000     4
```

Version 2014a on Linux (i7 processor)


8 tasks



Editor - /home/telzur/Documents/Teaching/BGU/PP/PP2015A/lectures/09/code/parallel0.m

parallel1.m x parallel0.m x +

```
1 - disp('Serial computation');
2 - ntasks=8;
3 - tic; for i=1:ntasks, c(:,i)=eig(rand(1000));
4 -   end; toc
5
6 - disp('parallel computation');
7 - delete(gcf);
8 - parpool('local');
9 - tic; parfor i=1:ntasks, d(:,i)=eig(rand(1000));
10 -   end; toc
11
```

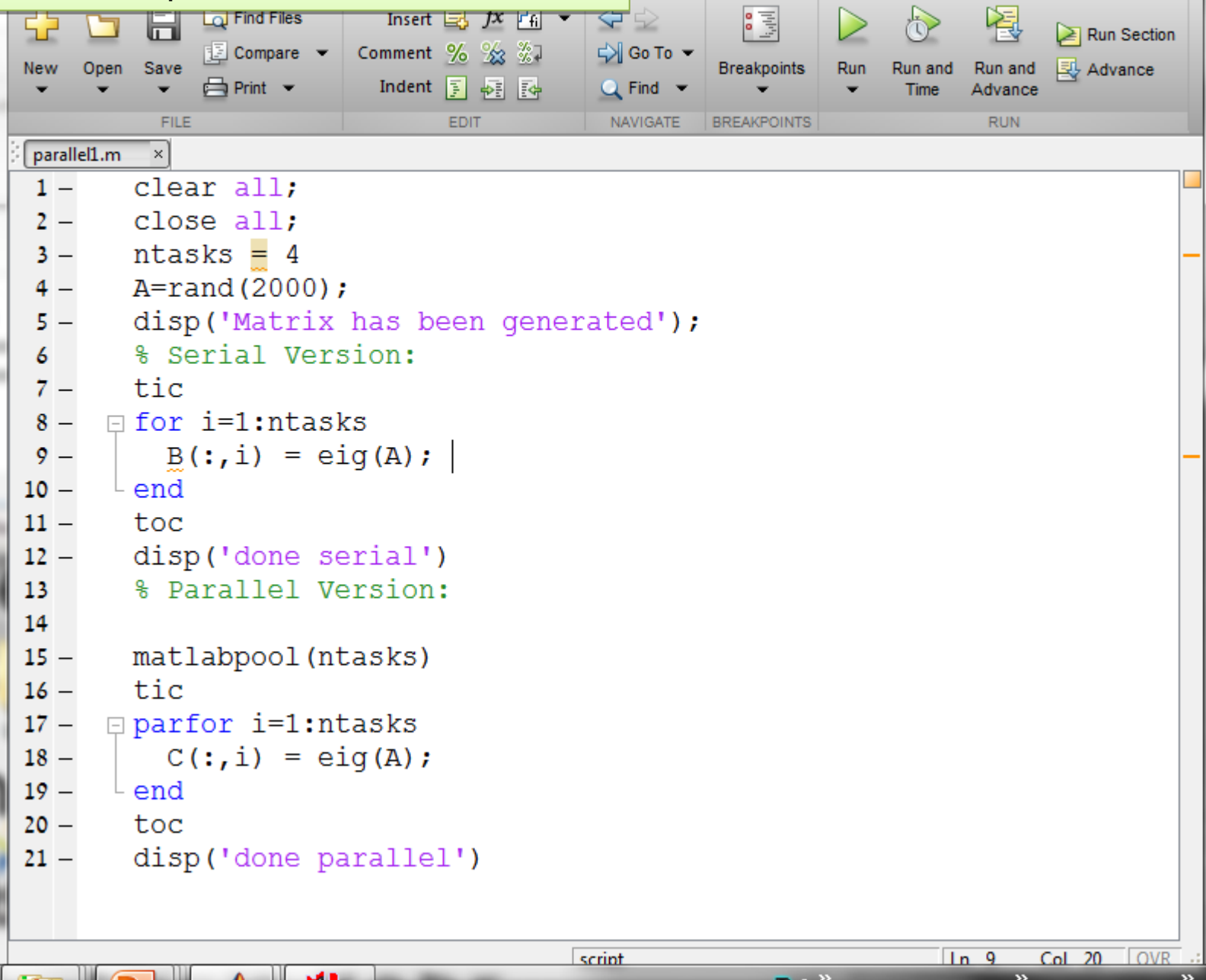


The screenshot shows the MATLAB Workspace window. It contains a table with two columns: 'Name' and 'Value'. The variables listed are:

Name	Value
ans	1x1 parallel
c	1000x8 complex
d	1000x8 complex
i	8
ntasks	8

```
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>>  
>> parallel0  
Serial computation  
Elapsed time is 7.800062 seconds.  
parallel computation  
Parallel pool using the 'local' profile is shutting down.  
Starting parallel pool (parpool) using the 'local' profile ... conr  
Elapsed time is 4.434908 seconds.  
fx >>
```

Demo: ~/lecture09/parallel1.m



The image shows a MATLAB IDE window with a script named 'parallel1.m'. The script compares the execution time of a serial loop versus a parallel 'parfor' loop for computing eigenvectors of a 2000x2000 matrix. The serial version uses a 'for' loop, while the parallel version uses 'parfor' after creating a MATLAB pool. The script includes comments and display messages to track progress and completion.

```
1 - clear all;
2 - close all;
3 - ntasks = 4
4 - A=rand(2000);
5 - disp('Matrix has been generated');
6 - % Serial Version:
7 - tic
8 - for i=1:ntasks
9 -     B(:,i) = eig(A);
10 - end
11 - toc
12 - disp('done serial')
13 - % Parallel Version:
14 -
15 - matlabpool(ntasks)
16 - tic
17 - parfor i=1:ntasks
18 -     C(:,i) = eig(A);
19 - end
20 - toc
21 - disp('done parallel')
```

The status bar at the bottom indicates the current position is Line 9, Column 20, and the file is named 'script'.

Profile Summary

Generated 26-May-2013 21:12:07 using cpu time

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
parallel1	1	56.691 s	28.421 s	
parallel_function	1	18.380 s	0.022 s	
parallel_function>distributed_execution	1	18.230 s	0.079 s	
...omp remoteparfor.getCompleteIntervals	3	18.125 s	0.038 s	
java.util.concurrent.LinkedBlockingQueue (Java method)	20	18.067 s	18.067 s	
matlabpool	1	9.890 s	0.076 s	
MatlabpoolHelper>MatlabpoolHelper.doOpen	1	9.152 s	0.018 s	
...lHelper>MatlabpoolHelper.doMatlabpool	1	9.152 s	0.000 s	
distcomp.interactiveclient.start	1	9.123 s	0.051 s	
distcomp.interactiveclient.pGetSockets	1	8.145 s	0.001 s	
...ient.pGetSockets>iGetSingleConnection	2	8.144 s	7.808 s	
...parseInputsAndCheckOutputsForFunction	1	0.649 s	0.001 s	
...atlabpoolHelper.parseMatlabpoolInputs	1	0.636 s	0.008 s	
...er>ProfileConfigHelper.buildScheduler	1	0.459 s	0.003 s	
parcluster	1	0.456 s	0.080 s	
...lusterAdaptor>iCreateCommunicatingJob	1	0.311 s	0.005 s	
Job.Job>Job.submit	1	0.308 s	-0.000 s	
...gJob>CJSSCommunicatingJob.submitOneJob	1	0.297 s	0.011 s	
Local.hSubmitCommunicatingJob	1	0.280 s	0.013 s	
Local.Local>Local.Local	1	0.209 s	0.018 s	
Cluster.createCommunicatingJob	1	0.184 s	0.002 s	
...hworks.toolbox.distcomp.pmode.Session (Java method)	25	0.174 s	0.174 s	
CJSSCluster>CJSSCluster.CJSSCluster	1	0.169 s	0.048 s	
etime	1036	0.169 s	0.169 s	
CJSSupport>CJSSupport.getProperties	31	0.153 s	0.012 s	
ProfileConfigHelper>iGetDefaultProfile	1	0.149 s	0.000 s	
...er>ProfileConfigHelper.getDefaultName	1	0.149 s	0.000 s	
...Helper>MatlabpoolHelper.checkConfigOk	1	0.149 s	0.000 s	
...oolHelper.checkConfigOk(profHelper,x)	1	0.149 s	0.000 s	
CJSJobMixIn>CJSJobMixIn.hGetProperty	26	0.146 s	0.002 s	

parallel1 (1 call, 56.691 sec)

Generated 26-May-2013 21:15:20 using cpu time.

script in file <C:\Users\telzun\Documents\BGU\Teaching\ParallelProcessing\PP2013B\lectures\09\parallel1.m>
[Copy to new window for comparing multiple runs](#)

This function changed during profiling or before generation of this report. Results may be incomplete or inaccurate




Refresh

- ☒ Show parent functions ☒ Show busy lines ☒ Show child functions
☒ Show Code Analyzer results ☒ Show file coverage ☒ Show function listing




Parents (calling functions)

No parent

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
9	B(:,i) = eig(A);	2	28.011 s	49.4%	
17	parfor i=1:ntasks	1	18.471 s	32.6%	
15	matlabpool(ntasks)	1	9.894 s	17.5%	
1	clear all;	1	0.195 s	0.3%	
4	A=rand(2000);	1	0.085 s	0.1%	
All other lines			0.035 s	0.1%	
Totals			56.691 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
parallel_function	function	1	18.380 s	32.4%	
matlabpool	function	1	9.890 s	17.4%	
parfor_endpoint_check	function	2	0 s	0%	
parfor_sliced_fcnhdl_check	function	1	0 s	0%	
close	function	1	0 s	0%	
Selftime (built-ins, overhead, etc.)			28.421 s	50.1%	
Totals			56.691 s	100%	

```
parallel1.m* x parallel0.m x +
1 - clear all;
2 - close all;
3 - delete(gcf);
4 - ntasks = 8;
5 - A=rand(2000);
6 - disp('Matrix has been generated');
7 - % Serial Version:
8 - tic
9 - for i=1:ntasks
10 -     B(:,i) = eig(A);
11 - end
12 - toc
13 - disp('done serial')
14 - % Parallel Version:
15 - matlabpool('local',ntasks)
16 - tic
17 - parfor i=1:ntasks
18 -     C(:,i) = eig(A);
19 - end
20 - toc
21 - disp('done parallel')
```

Parallel1.m
8 tasks on core i7

```
>> close all
>> clear all
>> parallel1
```

Parallel pool using the 'local' profile is shutting down.

```
ntasks =

      8
```

Matrix has been generated

Elapsed time is 31.812732 seconds.

done serial

Warning: matlabpool will be removed in a future release.

Use parpool instead.

Starting matlabpool using the 'local' profile ... connected to 8 workers.

Elapsed time is 19.150792 seconds.

done parallel

```
>>
```


Command Window

```
>>
>> spmd
R=rand(4);
end
>> R

R =

    Lab 1: class = double, size = [4  4]
    Lab 2: class = double, size = [4  4]
    Lab 3: class = double, size = [4  4]
    Lab 4: class = double, size = [4  4]

>> R{1}

ans =

    0.7324    0.6533    0.4855    0.4802
    0.8273    0.1966    0.1460    0.0414
    0.5708    0.0701    0.4374    0.0343
    0.0286    0.7294    0.8261    0.8813

>> R{2}

ans =

    0.3403    0.0352    0.1909    0.3058
    0.9855    0.4162    0.4921    0.1040
    0.1635    0.8936    0.7235    0.4481
    0.1263    0.6937    0.8305    0.3101

>> R{3}

ans =



    0.3499    0.5849    0.2805    0.6119
    0.5549    0.0615    0.6036    0.1658
    0.8473    0.8638    0.1458    0.9970
    0.1063    0.5371    0.7924    0.0217


>> R{4}

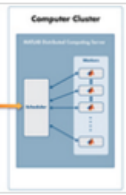

ans =

    0.3395    0.9410    0.4371    0.4194
    0.8902    0.5437    0.6271    0.9787
    0.8345    0.4181    0.4030    0.5508
    0.8874    0.6705    0.9059    0.8437
```

optional



Search for add-ons




Installed


Tutorials: Parallel and GPU Computing with MATLAB: All in one (9 parts)

version 1.5.0.1 (12.7 KB) by [MathWorks Parallel Computing Toolbox Team](#) **STAFF**

Tutorials on Parallel and GPU Computing with MATLAB

 Collection

★★★★★ 1 Rating

18 Downloads 

Updated 1 Sep 2016

[View License](#)

Open Folder

Manage

Overview


Functions

This submission contains all code examples used in tutorial series for Parallel and GPU Computing with MATLAB available here:
<http://www.mathworks.com/products/parallel-computing/tutorials.html>

Topics covered:

1. Product Landscape (no code examples)
2. Prerequisites and Setup (no code examples)
3. Quick Success with parfor
4. Deeper Insights into Using parfor
5. Batch Processing
6. Scaling to Clusters
7. spmd - Parallel Code Beyond parfor
8. Distributed Arrays
9. GPU Computing with MATLAB

Requires

 [Parallel Computing Toolbox](#)

A NVIDIA CUDA GPU with compute capability 2.0 or above is required for running GPU computing example code

MATLAB Release Compatibility

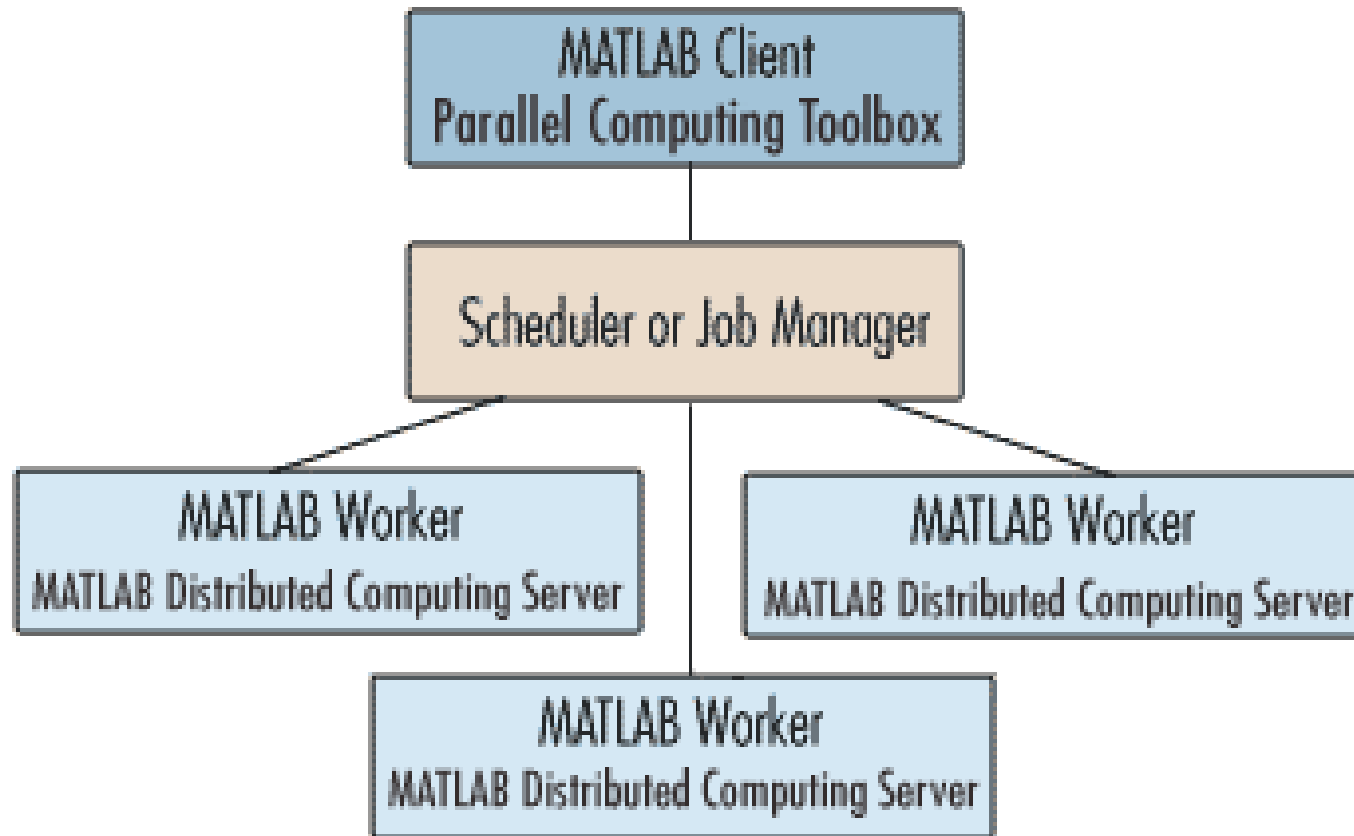
Created with R2014b
Compatible with any release

Platform Compatibility

☒ Windows ☒ macOS ☒ Linux

<https://www.mathworks.com/videos/series/parallel-and-gpu-computing-tutorials-97719.html>

Parallel Computing Toolbox and MATLAB Distributed Computing



Parallel Computing with Matlab on Amazon Cloud

MATLAB Parallel Computing Tools: Basic Setup and Requirements

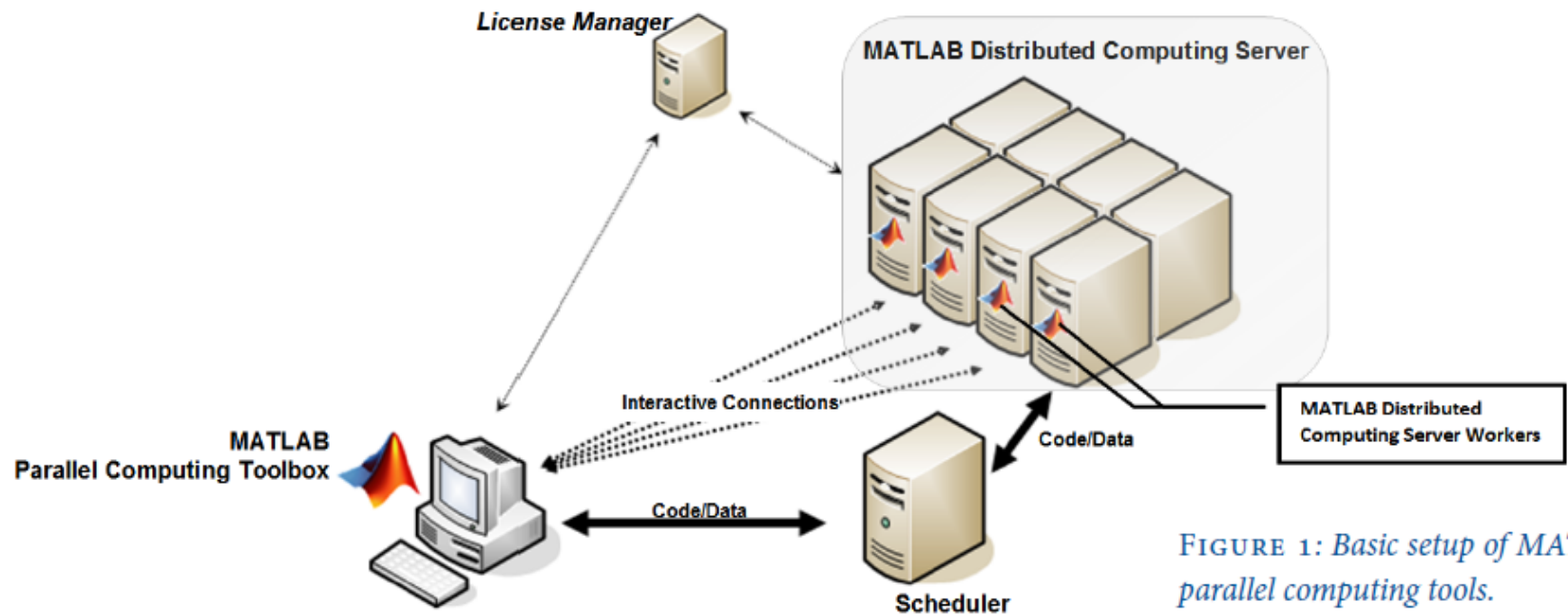


FIGURE 1: Basic setup of MATLAB parallel computing tools.

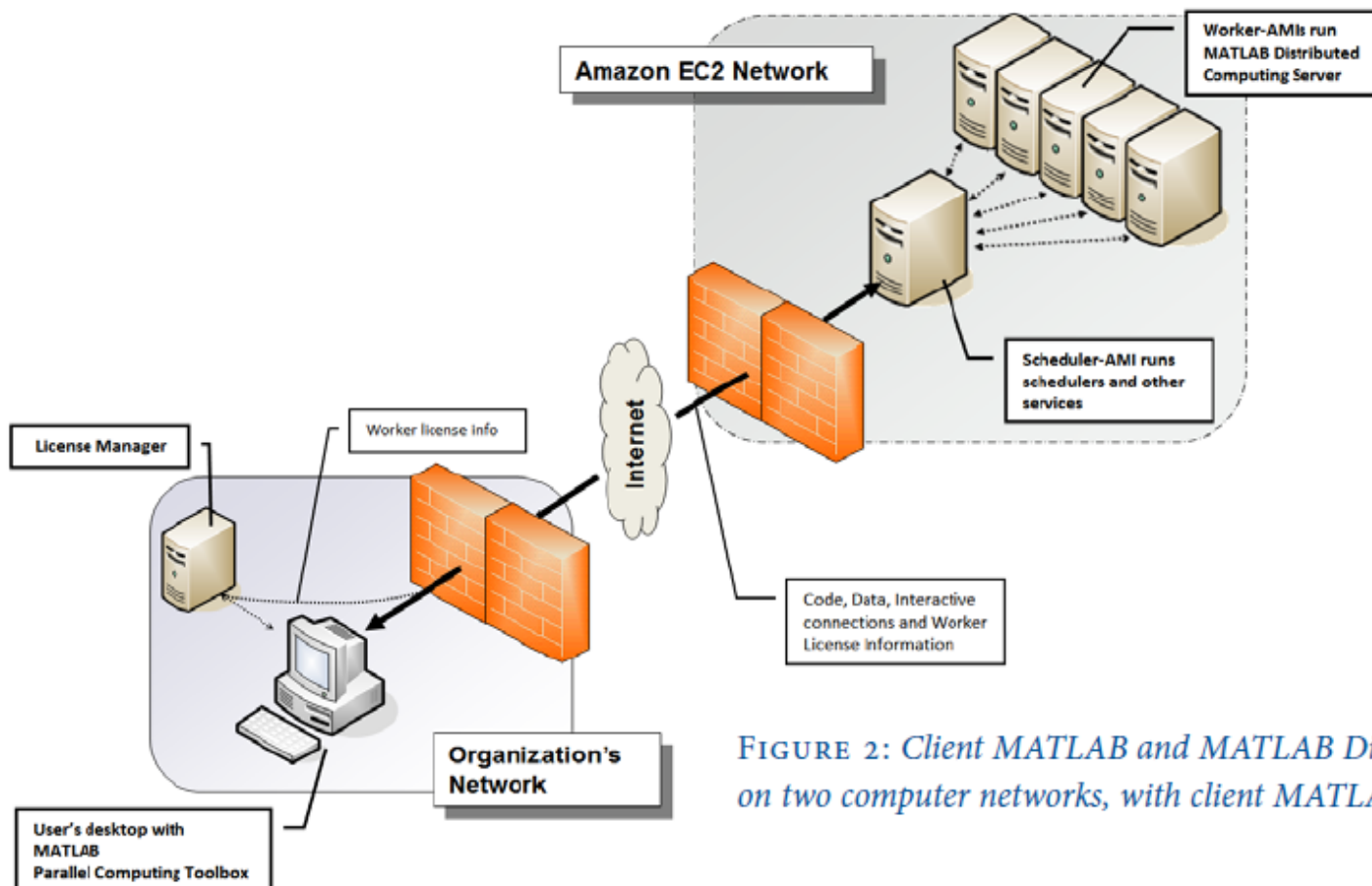


FIGURE 2: Client MATLAB and MATLAB Distributed Computing Server on two computer networks, with client MATLAB on a user's desktop.

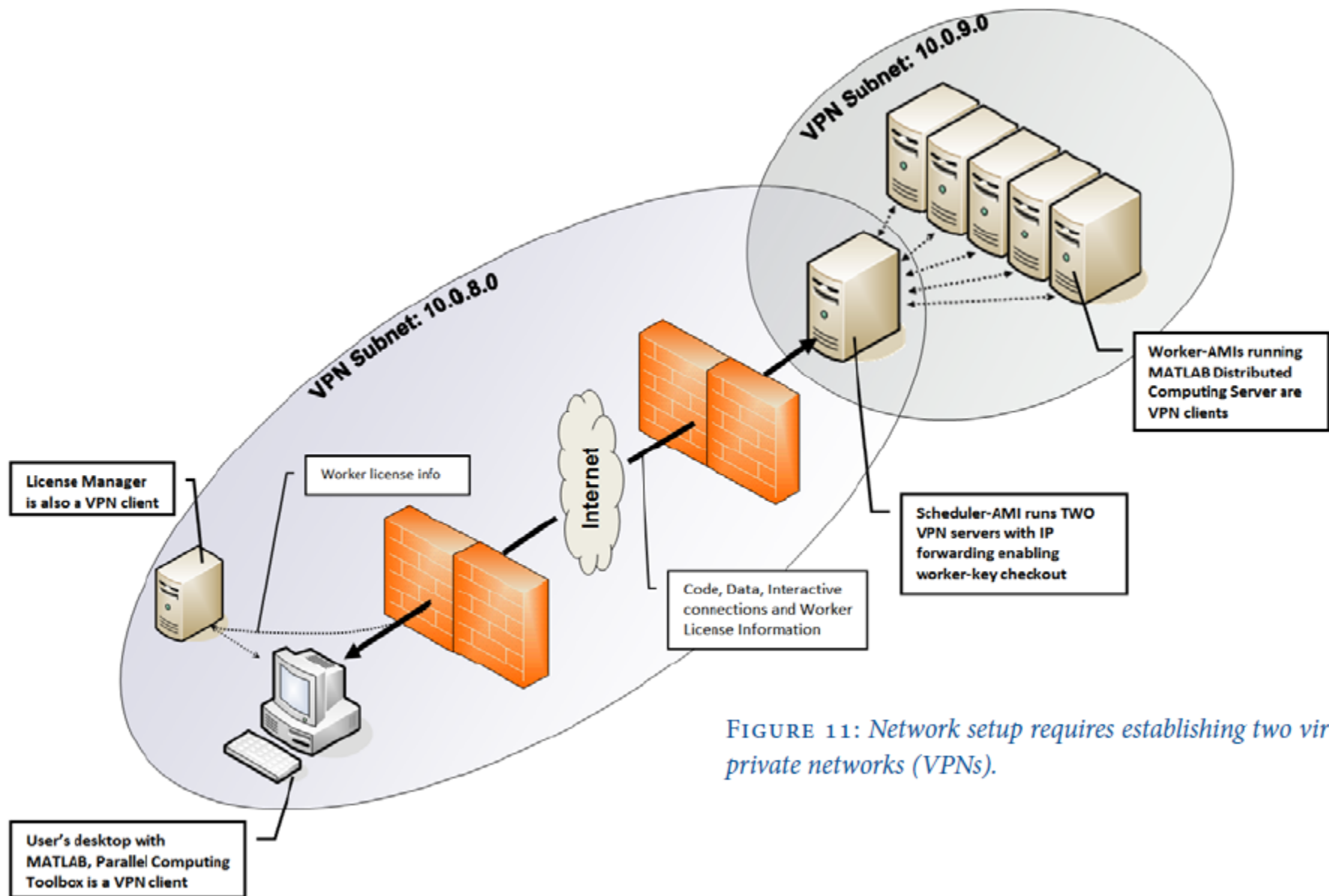


FIGURE 11: Network setup requires establishing two virtual private networks (VPNs).

Matlab and GPU computing

`~/.../lectures/08/matlab_code`

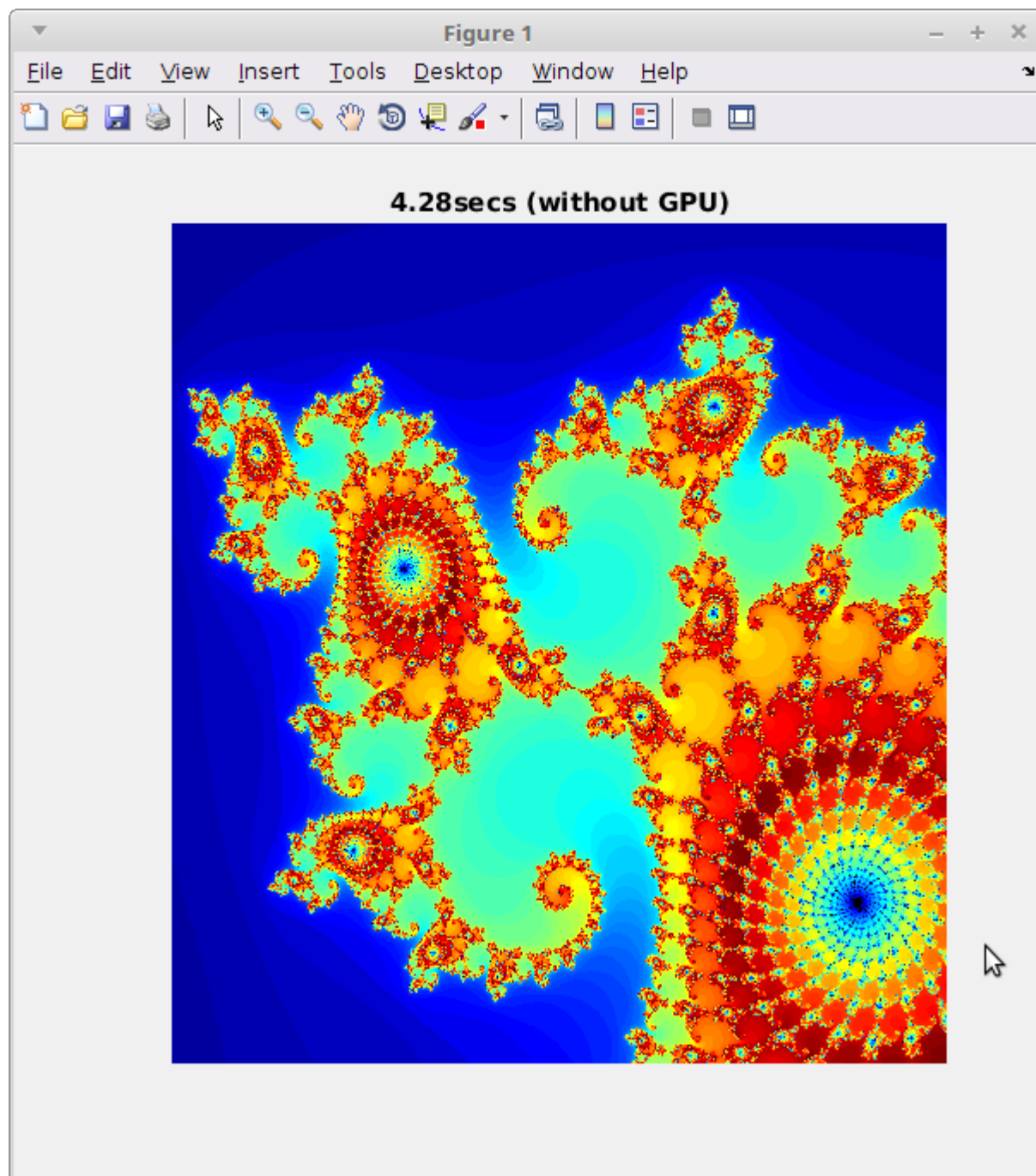


Figure 2

File Edit View Insert Tools Desktop Window Help

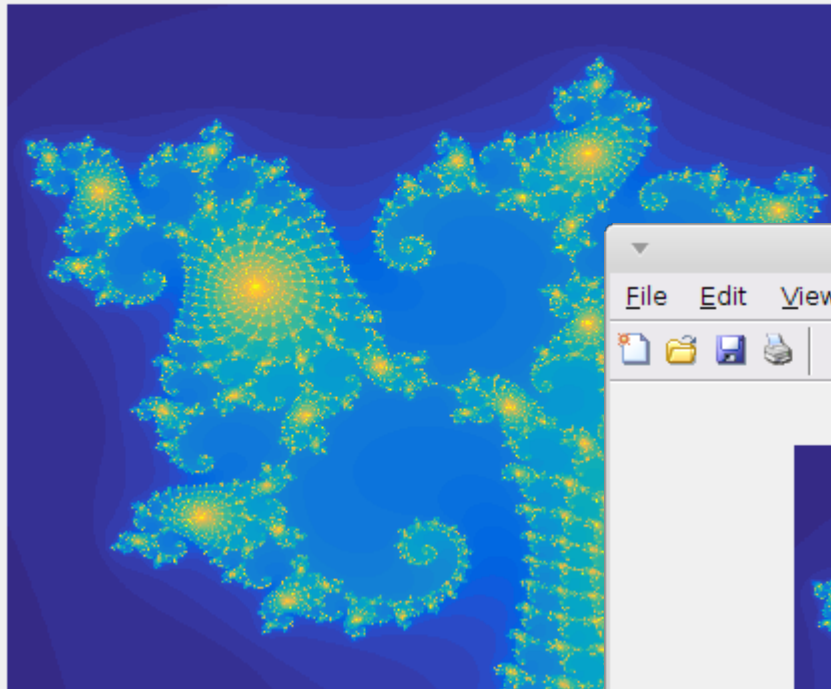
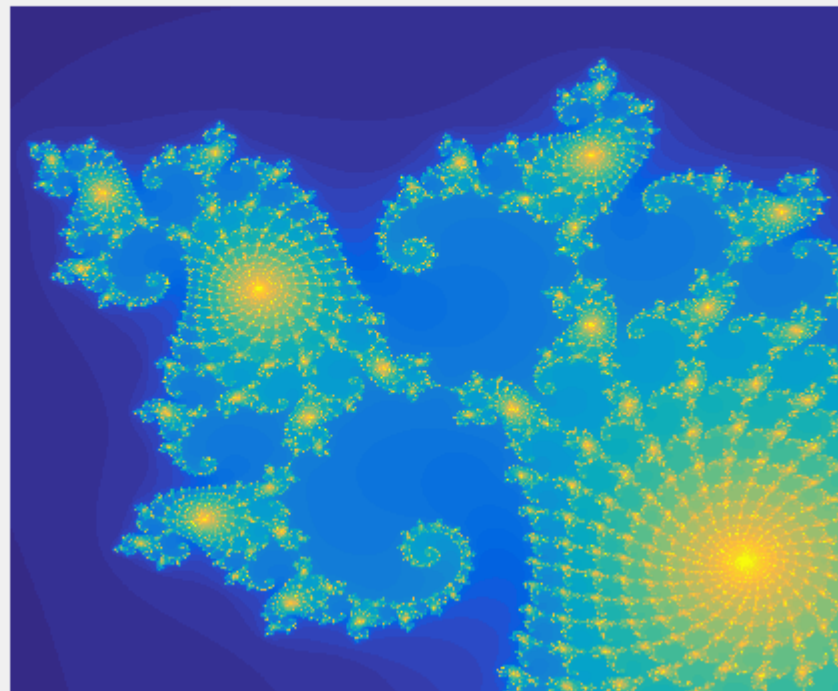
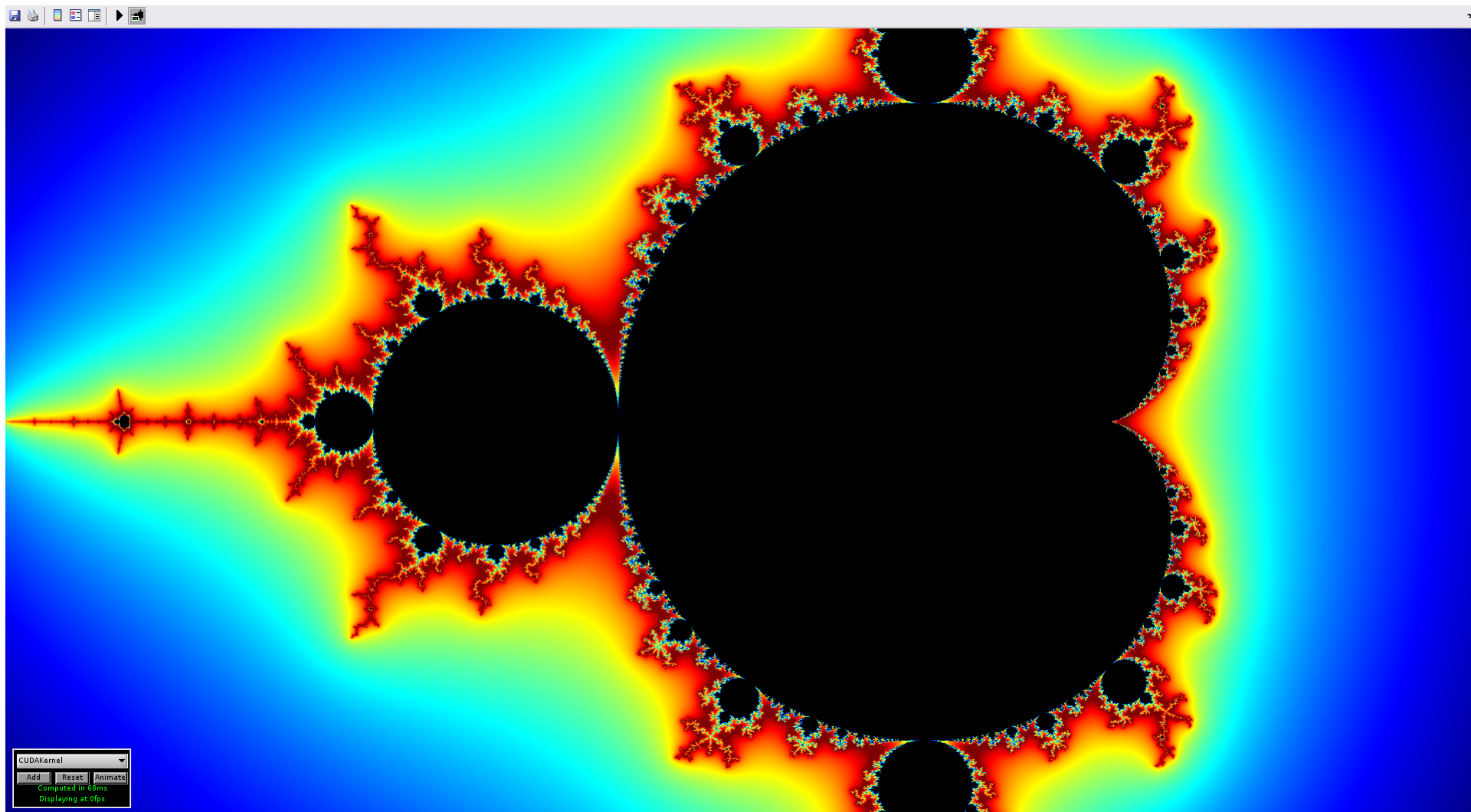
**3.151secs (naive GPU) = 1.4x faster**

Figure 3

File Edit View Insert Tools Desktop Window Help

**0.157secs (GPU CUDAKernel) = 27.2x faster**

MandelbrotViewer



Let's try this

```
A = rand(100, GPUsingle); % A is on GPU memory  
B = rand(100, GPUsingle); % B is on GPU memory  
C = A+B; % executed on GPU.  
D = fft(C); % executed on GPU
```

Executed on GPU

```
A = single(rand(100)); % A is on CPU memory  
B = single(rand(100)); % B is on CPU memory  
C = A+B; % executed on CPU.  
D = fft(C); % executed on CPU
```

Executed on CPU

Matlab Parallel Addons

Contribute | Manage Add-Ons

Clear Filters x Search for add-ons



Filter by Source

- ☐ MathWorks 5
- ☐ Community 131

Filter by Category

< Clear Categories

Using MATLAB

- Language Fundamentals 879
- Data Import and Analysis 998
- Mathematics 1,392
- Graphics 1,858
- Programming 367
- App Building 409
- Software Development Tools 146
- External Language Interfaces 432
- Environment and Settings 120
- Installation, Licensing, and Activation 10
- Parallel Computing 136**
- Parallel Computing 99
- MATLAB Parallel Server 17
- Application Deployment 62

136 RESULTS

Parallel Computing (136)



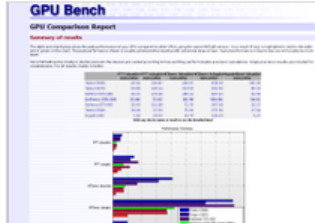
Installed

Parallel Computing Toolbox

Perform parallel computations on multicore computers, GPUs, and clusters




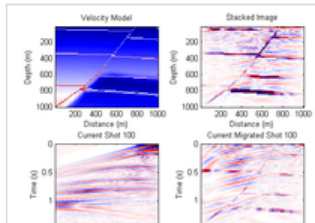
Progress monitor (progress bar)



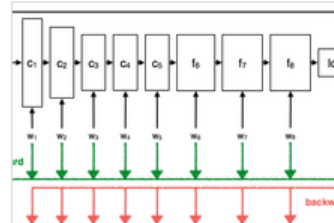
GPUBench

Compare GPUs using standard numerical benchmarks in MATLAB.

192 Downloads 





Large Data in MATLAB: A Sciencia Data Processing



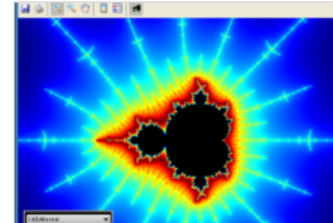
vlfate/matconvnet

MatConvNet: CNNs for MATLAB

48 Downloads 




Parallel Computing Toolbox



A GPU Mandelbrot Set

Explore the Mandelbrot Set using MATLAB and a GPU.

44 Downloads 

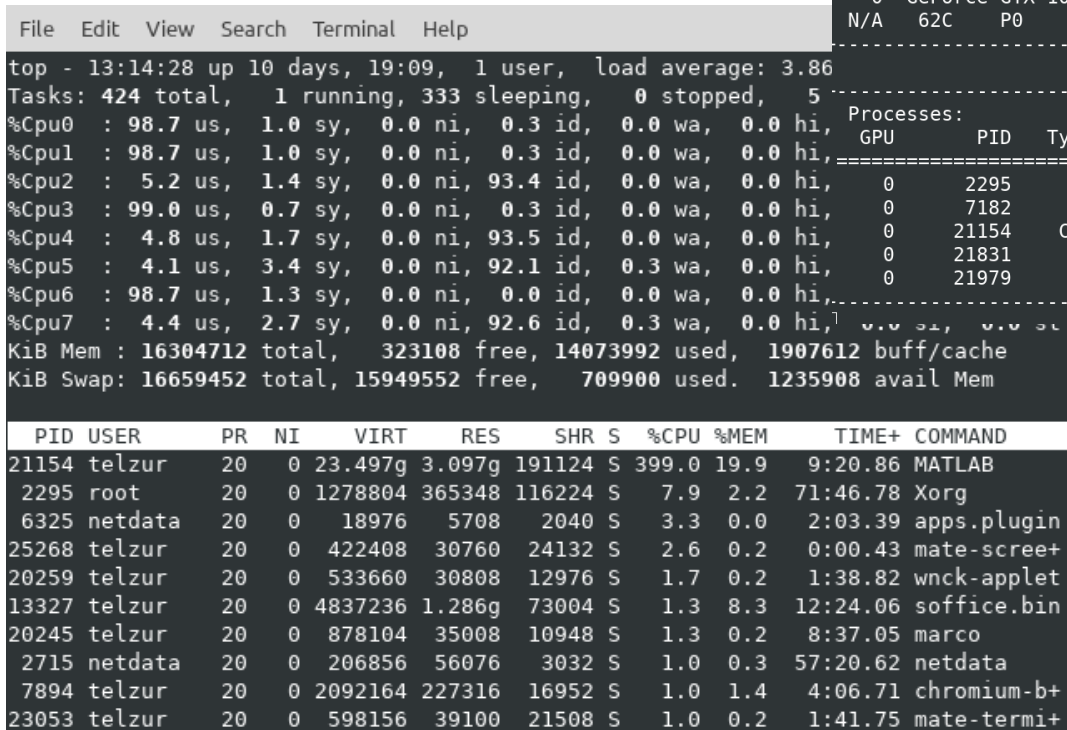
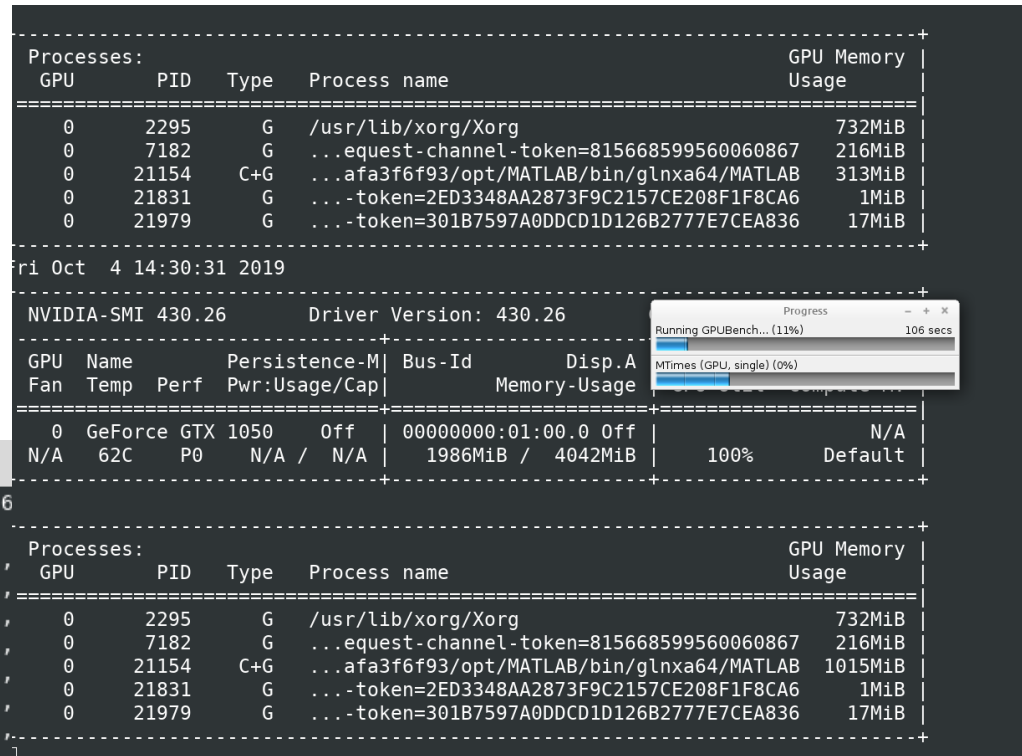
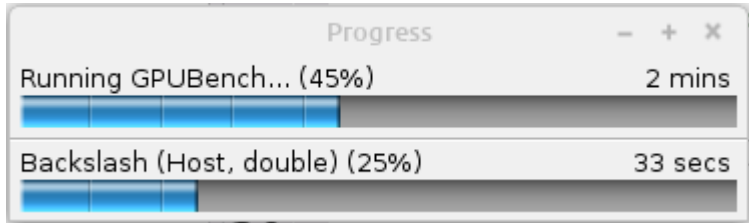


LYNX

FAST MACHINE LEARNING DEVELOPMENT

Lynx MATLAB Toolbox

gpuBench



GPU Comparison Report: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

Summary of results

The table and chart below show the peak performance of various GPUs using the same MATLAB version. Your results (if any) are highlighted in bold in the table and on the chart. All other results are from pre-stored data. The peak performance shown is usually achieved when dealing with extremely large arrays. Typical performance in day-to-day use will usually be much lower.

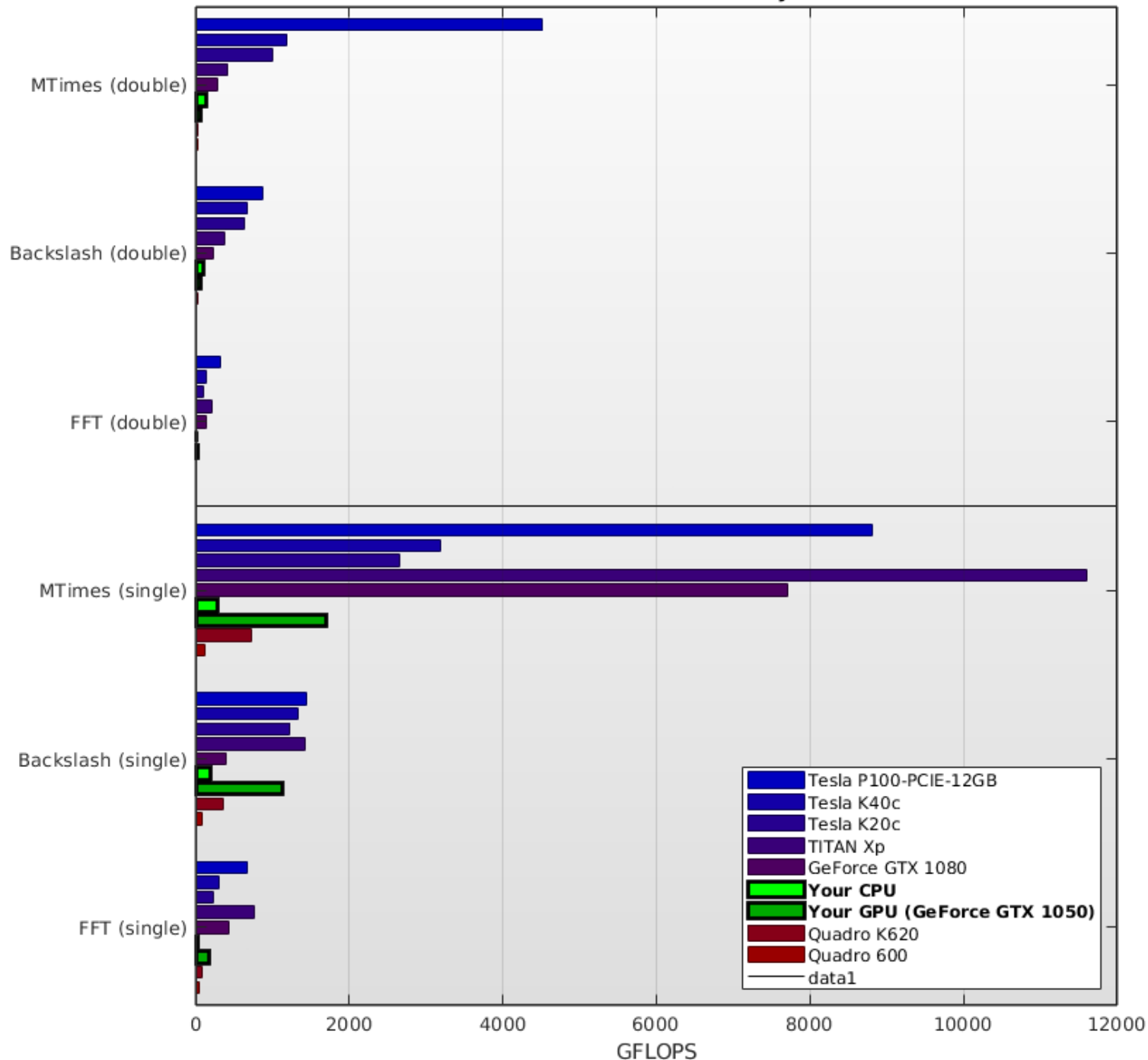
Results captured using the CPUs on the host PC (i.e. without using a GPU) are included for comparison.

Since MATLAB works mostly in double precision the devices are ranked according to how well they perform double-precision calculations. Single precision results are included for completeness. For all results, higher is better.

	Results for data-type 'double' (In GFLOPS)			Results for data-type 'single' (In GFLOPS)		
	MTimes	Backslash	FFT	MTimes	Backslash	FFT
Tesla P100-PCIE-12GB	4518.23	878.97	313.43	8807.20	1439.15	676.20
Tesla K40c	1189.54	677.12	135.88	3187.76	1334.17	294.86
Tesla K20c	1004.06	641.42	106.09	2657.01	1230.28	235.20
TITAN Xp	422.47	371.37	207.24	11607.69	1426.76	763.56
GeForce GTX 1080	280.84	223.05	137.66	7707.01	399.37	424.60
Your CPU	137.45	96.16	14.72	285.93	199.74	21.16
Your GPU (GeForce GTX 1050)	61.24	55.61	31.65	1699.35	1131.12	181.04
Quadro K620	25.45	22.77	12.75	716.71	350.31	75.00
Quadro 600	19.71	17.55	7.62	117.99	88.64	38.58

(click any device name or result to see the detailed data)

Performance Summary



Results for Backslash (double)

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	2.94	0.01
4,096	180,907	1.22	0.15
16,384	1,422,677	1.02	1.40
65,536	11,283,115	4.17	2.71
262,144	89,871,701	5.41	16.62
1,048,576	717,400,747	18.84	38.07
4,194,304	5,732,914,517	103.10	55.61

Backslash (double)

GFLOPS (higher is better)

Number of elements

Legend:

- Tesla P100-PCIE-12GB
- Tesla K40c
- Tesla K20c
- TITAN Xp
- GeForce GTX 1080
- Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- GeForce GTX 1050 (selected)**
- Quadro K620
- Quadro 600

Number of elements	Tesla P100-PCIE-12GB	Tesla K40c	Tesla K20c	TITAN Xp	GeForce GTX 1080	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz	GeForce GTX 1050 (selected)	Quadro K620	Quadro 600
10 ³	0	0	0	0	0	0	0	0	0
10 ⁴	0	0	0	0	0	0	0	0	0
10 ⁵	0	0	0	0	0	20	0	0	0
10 ⁶	0	0	0	0	0	50	40	0	0
4.5 × 10 ⁶	0	0	0	0	0	70	60	0	0
10 ⁷	0	0	0	0	0	80	0	0	0
10 ⁸	0	0	0	0	0	100	0	0	0

Results for Backslash (single)

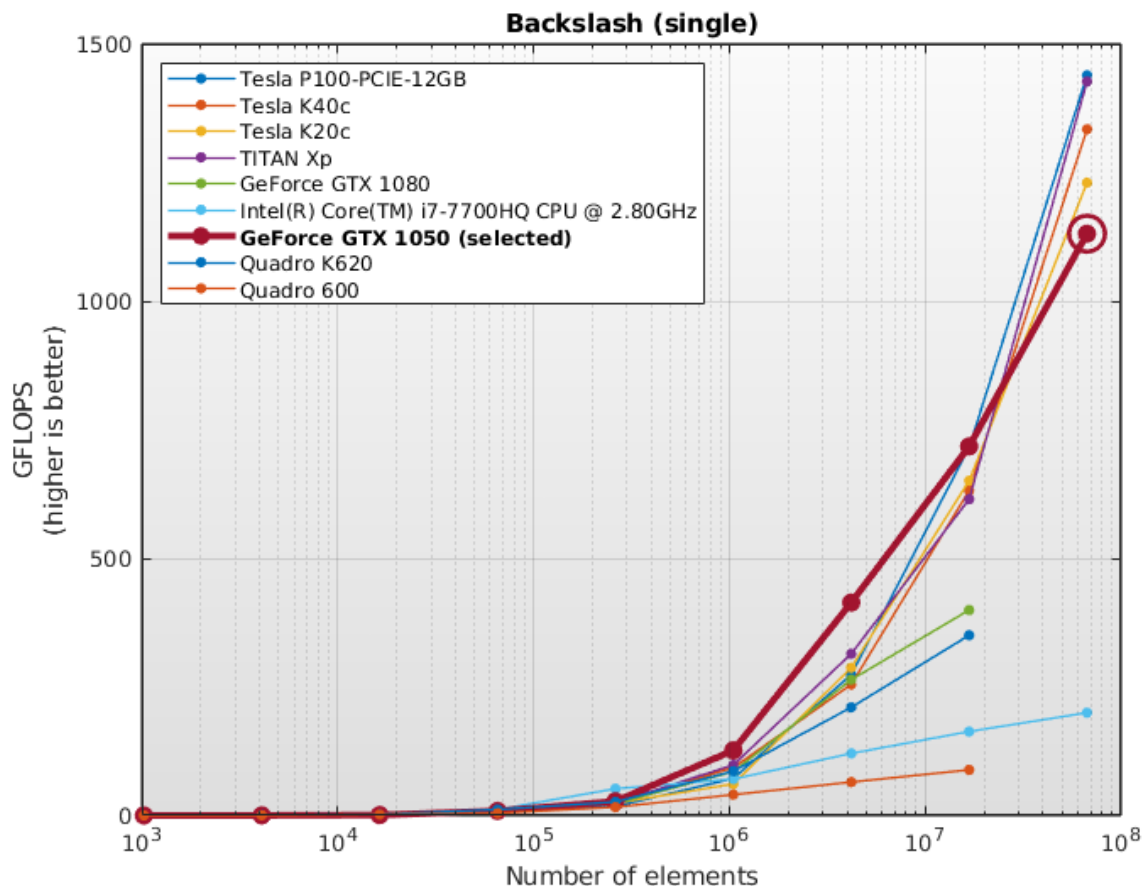
These results show the performance of the GPU or host PC when calculating the **matrix left division** of an $N \times N$ matrix with an $N \times 1$ vector. The number of operations is assumed to be $\frac{2}{3} * N^3 + \frac{3}{2} * N^2$.

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Raw data for GeForce GTX 1050 - Backslash (single)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	3.47	0.01
4,096	180,907	1.84	0.10
16,384	1,422,677	1.66	0.86
65,536	11,283,115	1.28	8.79
262,144	89,871,701	3.27	27.45
1,048,576	717,400,747	5.67	126.57
4,194,304	5,732,914,517	13.85	413.83
16,777,216	45,838,150,315	63.88	717.55
67,108,864	366,604,539,221	324.11	1131.12

(N gigaflops = $N \times 10^9$ operations per second)



Results for FFT (double)

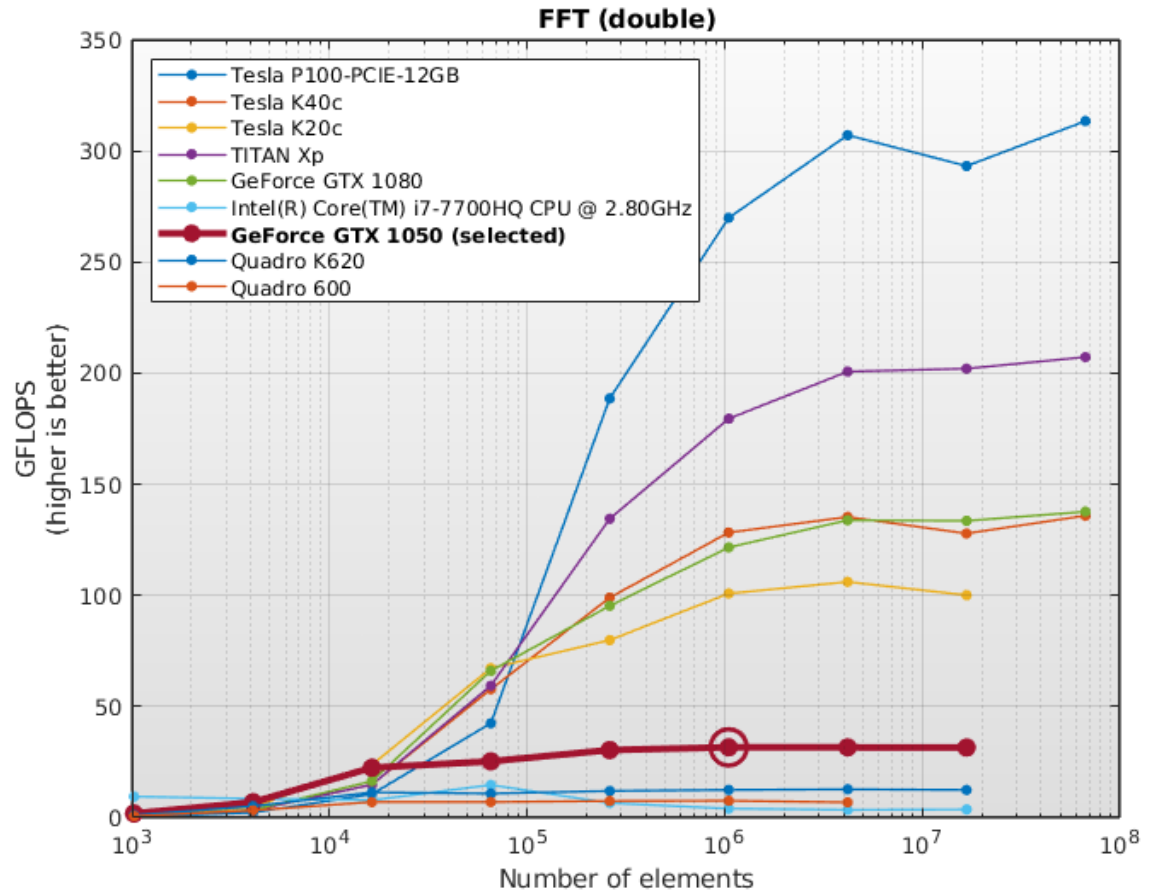
These results show the performance of the GPU or host PC when calculating the **Fast-Fourier-Transform** of a vector of complex numbers. The number of operations for a vector of length N is assumed to be $5 \cdot N \cdot \log_2(N)$.

This calculation is usually memory-bound, i.e. the performance depends mainly on how fast the GPU or host PC can read and write data.

Raw data for GeForce GTX 1050 - FFT (double)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	51,200	0.03	1.87
4,096	245,760	0.04	6.92
16,384	1,146,880	0.05	22.47
65,536	5,242,880	0.21	25.33
262,144	23,592,960	0.78	30.42
1,048,576	104,857,600	3.31	31.65
4,194,304	461,373,440	14.59	31.62
16,777,216	2,013,265,920	63.81	31.55

(N gigaflops = $N \times 10^9$ operations per second)



Results for FFT (single)

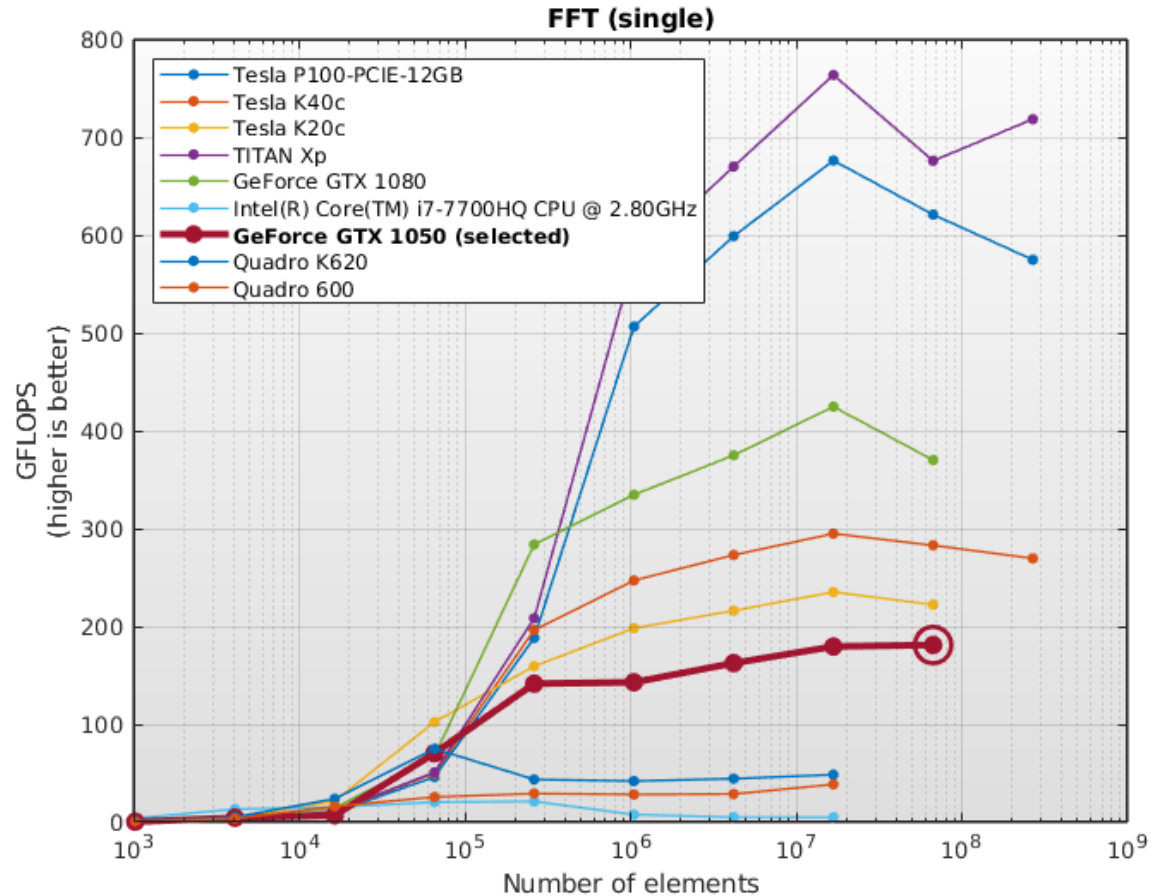
These results show the performance of the GPU or host PC when calculating the [Fast-Fourier-Transform](#) of a vector of complex numbers. The number of operations for a vector of length N is assumed to be $5 \cdot N \cdot \log_2(N)$.

This calculation is usually memory-bound, i.e. the performance depends mainly on how fast the GPU or host PC can read and write data.

Raw data for GeForce GTX 1050 - FFT (single)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	51,200	0.13	0.41
4,096	245,760	0.06	4.44
16,384	1,146,880	0.17	6.90
65,536	5,242,880	0.07	70.38
262,144	23,592,960	0.17	141.56
1,048,576	104,857,600	0.73	142.94
4,194,304	461,373,440	2.84	162.50
16,777,216	2,013,265,920	11.22	179.50
67,108,864	8,724,152,320	48.19	181.04

(N gigaflops = $N \times 10^9$ operations per second)



CPU results

GPU Performance Details: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

- Contents:**
- System Configuration
 - Results for datatype double
 - MTimes (double)
 - Backslash (double)
 - FFT (double)
 - Results for datatype single
 - MTimes (single)
 - Backslash (single)
 - FFT (single)

System Configuration

MATLAB Release: R2019b

Host

Name	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Clock	900.349 MHz
Cache	6144 KB
NumProcessors	4
OSType	Linux
OSVersion	buildd@lgw01-amd64-031

Results for Backslash (double)

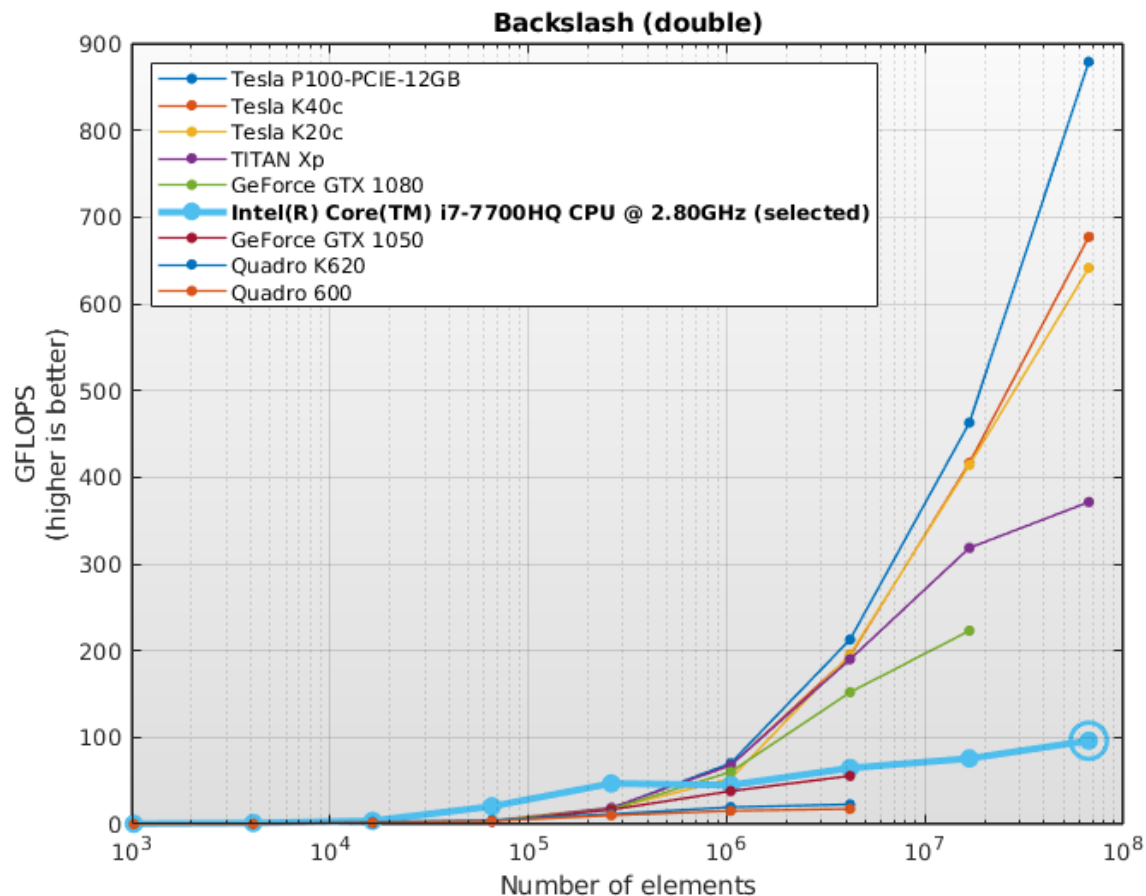
These results show the performance of the GPU or host PC when calculating the **matrix left division** of an $N \times N$ matrix with an $N \times 1$ vector. The number of operations is assumed to be $\frac{2}{3}N^3 + \frac{3}{2}N^2$.

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Raw data for Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz - Backslash (double)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	0.05	0.45
4,096	180,907	0.14	1.30
16,384	1,422,677	0.37	3.81
65,536	11,283,115	0.55	20.47
262,144	89,871,701	1.91	46.94
1,048,576	717,400,747	15.93	45.02
4,194,304	5,732,914,517	88.67	64.65
16,777,216	45,838,150,315	605.55	75.70
67,108,864	366,604,539,221	3812.38	96.16

(N gigaflops = $N \times 10^9$ operations per second)



MatlabMPI and pMatlab

Parallel Matlab (Octave) using MatlabMPI

Files location: vdwarf - /usr/local/PP/MatlabMPI

Read the README there!


cd to the **examples** directory

```
eval( MPI_Run('basic', 3,machines) );  
where:  
machines = {'vdwarf1' 'vdwarf2' 'vdwrf3'}
```

MatlabMPI

<http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html#introduction>

Exit full screen (F11)

 **LINCOLN LABORATORY**
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Home | Contact Us | Sitemap

SEARCH

About > | Mission Areas > | Employment > | College Recruiting > | News > | Publications > | Outreach > | Workshops/Education >

Home > Mission Areas > ISR Systems and Technology > MatlabMPI

MATLABMPI

Space Control >

Air and Missile Defense Technology >

Communications and Information Technology >

ISR Systems and Technology >

- MatlabMPI
- pMatlab
- HPEC Challenge

Advanced Electronics Technology >

Tactical Systems >

Homeland Protection >

Air Traffic Control >

Parallel Programming with MatlabMPI

Dr. Jeremy Kepner
kepner@ll.mit.edu

I. INTRODUCTION

Matlab is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing and system evaluation. Many of these computations could benefit from faster execution on a parallel computer. There have been many previous attempts to provide an efficient mechanism for running Matlab programs on parallel computers. These efforts have faced numerous challenges and none have received widespread acceptance.

In the world of parallel computing the Message Passing Interface (MPI) is the de facto standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard Matlab file i/o, resulting in a "pure" Matlab implementation that is exceedingly small (~300 lines of code). Thus, MatlabMPI will run on any combination of computers that Matlab supports. In addition, because of its small size, it is simple to download and use (and modify if you like).

MatlabMPI Page Contents

- [Introduction](#)
- [Download](#)
- [Requirements](#)
- [Installing and Running](#)
- [Launching and File I/O](#)
- [Error Handling](#)
- [Running on Linux](#)
- [Running on MacOSX](#)
- [Running on PC](#)
- [Other Optimizations](#)
- [Running in Batch Mode](#)
- [Other Settings](#)
- [Diagnostics and Troubleshooting](#)
- [First-Time User's Rules of Thumb](#)
- [Files](#)

pMatlab: Parallel Matlab Toolbox

pMatlab provides a set of Matlab data structures and functions that implement distributed Matlab arrays

[to pMatlab page >](#)

uting.iso

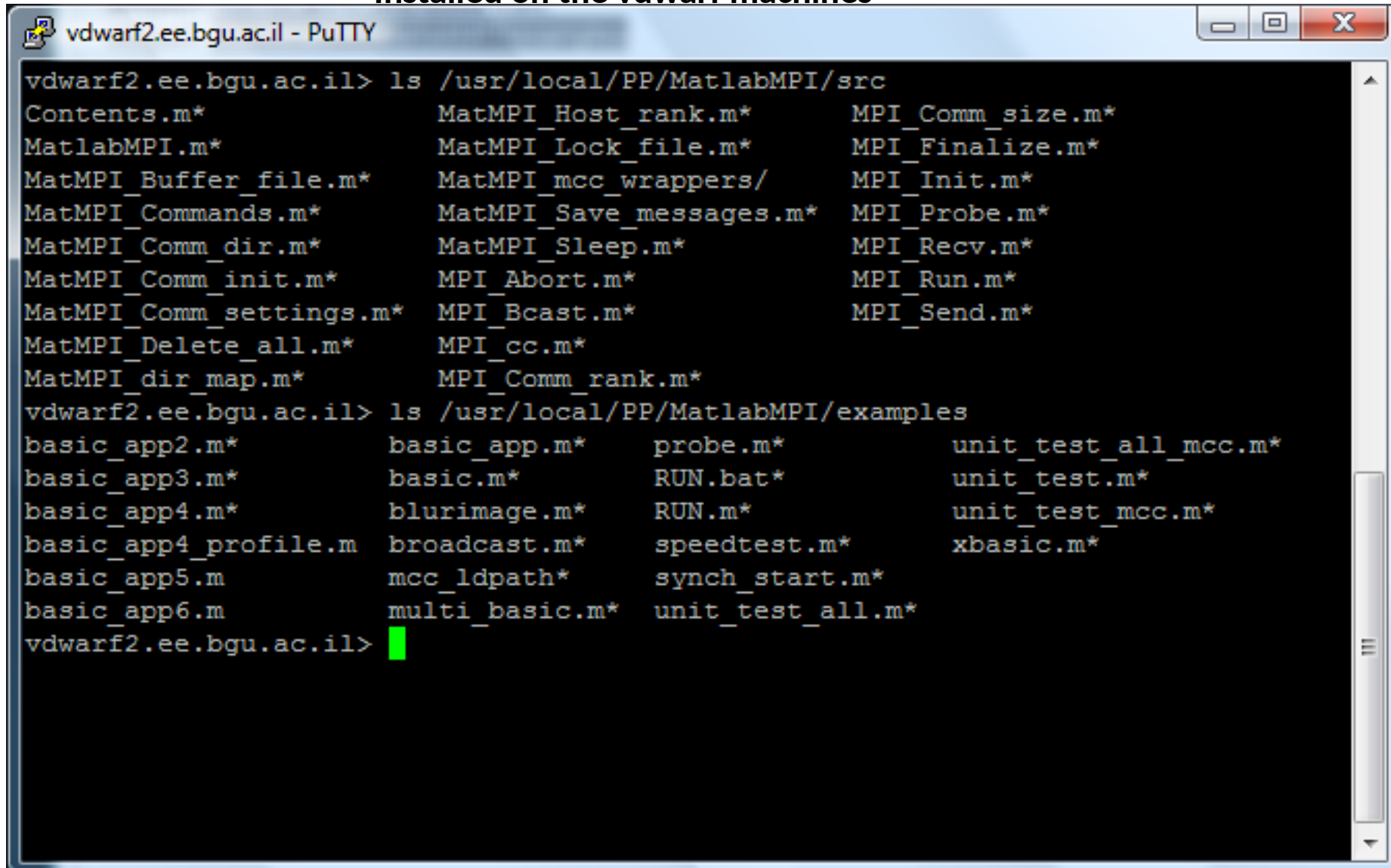
Show all downloads...

Available examples:

xbasic.m	Extremely simple MatlabMPI program that prints out the rank of each processor.
basic.m	Simple MatlabMPI program that sends data from processor 1 to processor 0.
multi_basic.m	Simple MatlabMPI program that sends data from processor 1 to processor 0 a few times.
probe.m	Simple MatlabMPI program that demonstrates the using MPI_Probe to check for incoming messages.
broadcast.m	Tests MatlabMPI broadcast command.
basic_app.m	Examples of the most common usages of MatlabMPI.
basic_app2.m	Examples of the most common usages of MatlabMPI.
basic_app3.m	Examples of the most common usages of MatlabMPI.
basic_app4.m	Examples of the most common usages of MatlabMPI.
blurimage.m	MatlabMPI test parallel image processing application.
speedtest.m	Times MatlabMPI for a variety of messages.
synch_start.m	Function for synchronizing starts.
machines.m	Example script for creating a machine description.
unit_test.m	Wrapper for using an example as a unit test.
unit_test_all.m	Calls all of the examples as way of testing the entire library.
unit_test_mcc.m	Wrapper for using an example as a mcc unit test.
unit_test_all_mcc.m	Calls all of the examples using MPI_cc as way of testing the entire library.

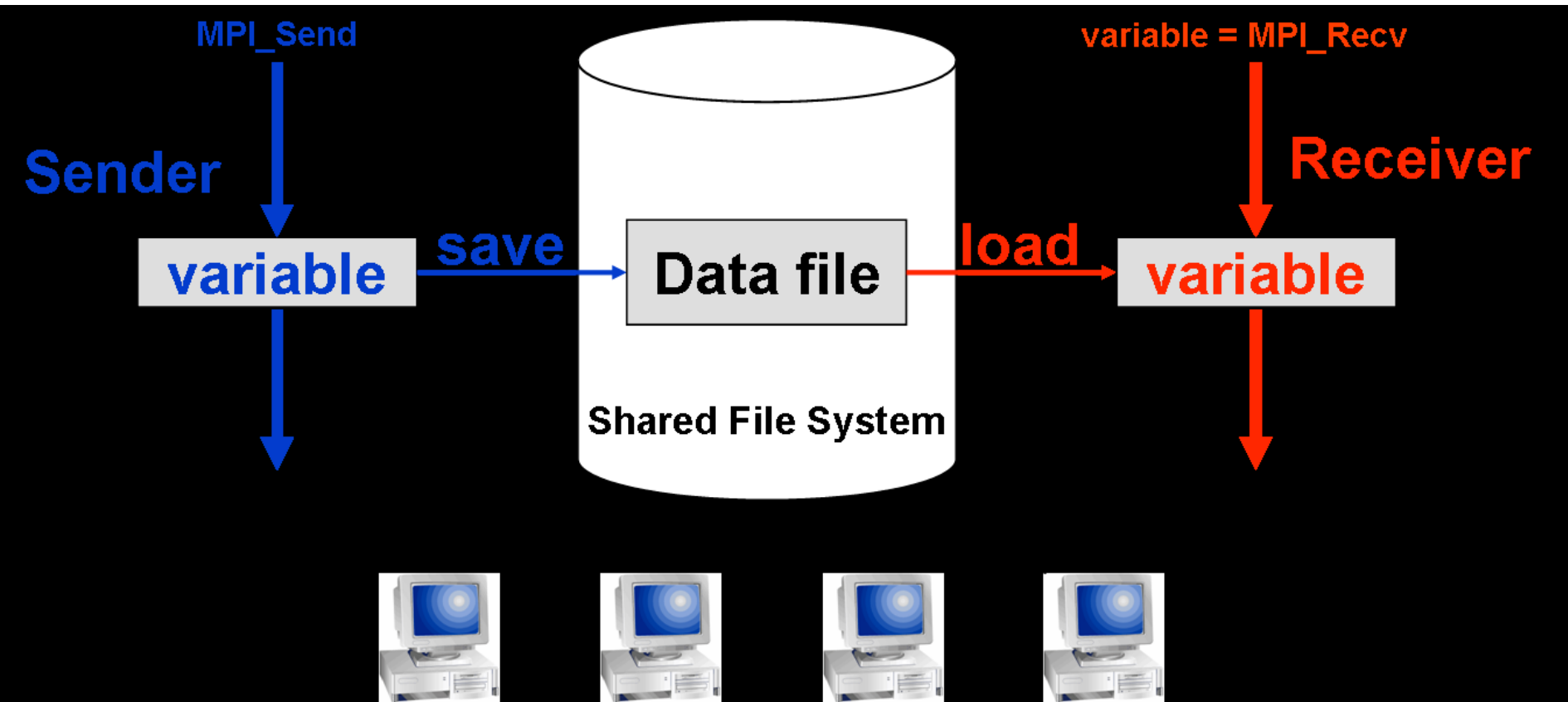
MatlabMPI Demo

Installed on the vdwarf machines



```
vdwarf2.ee.bgu.ac.il - PuTTY
vdwarf2.ee.bgu.ac.il> ls /usr/local/PP/MatlabMPI/src
Contents.m*           MatMPI_Host_rank.m*   MPI_Comm_size.m*
MatlabMPI.m*          MatMPI_Lock_file.m*   MPI_Finalize.m*
MatMPI_Buffer_file.m* MatMPI_mcc_wrappers/  MPI_Init.m*
MatMPI_Commands.m*    MatMPI_Save_messages.m* MPI_Probe.m*
MatMPI_Comm_dir.m*    MatMPI_Sleep.m*       MPI_Recv.m*
MatMPI_Comm_init.m*   MPI_Abort.m*          MPI_Run.m*
MatMPI_Comm_settings.m* MPI_Bcast.m*          MPI_Send.m*
MatMPI_Delete_all.m*  MPI_cc.m*
MatMPI_dir_map.m*     MPI_Comm_rank.m*
vdwarf2.ee.bgu.ac.il> ls /usr/local/PP/MatlabMPI/examples
basic_app2.m*      basic_app.m*      probe.m*          unit_test_all_mcc.m*
basic_app3.m*      basic.m*          RUN.bat*          unit_test.m*
basic_app4.m*      blurimage.m*      RUN.m*            unit_test_mcc.m*
basic_app4_profile.m broadcast.m*        speedtest.m*       xbasic.m*
basic_app5.m       mcc_ldpath*       synch_start.m*
basic_app6.m       multi_basic.m*    unit_test_all.m*
vdwarf2.ee.bgu.ac.il> █
```


MatlabMPI implements the fundamental communication operations in MPI using MATLAB's file I/O functions.



MatlabMPI

<http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html#introduction>

Exit full screen (F11)

 **LINCOLN LABORATORY**
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Home | Contact Us | Sitemap

SEARCH

About > | Mission Areas > | Employment > | College Recruiting > | News > | Publications > | Outreach > | Workshops/Education >

Home > Mission Areas > ISR Systems and Technology > MatlabMPI

MATLABMPI

Space Control >

Air and Missile Defense Technology >

Communications and Information Technology >

ISR Systems and Technology >

- MatlabMPI
- pMatlab
- HPEC Challenge

Advanced Electronics Technology >

Tactical Systems >

Homeland Protection >

Air Traffic Control >

Parallel Programming with MatlabMPI

Dr. Jeremy Kepner
kepner@ll.mit.edu

I. INTRODUCTION

Matlab is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing and system evaluation. Many of these computations could benefit from faster execution on a parallel computer. There have been many previous attempts to provide an efficient mechanism for running Matlab programs on parallel computers. These efforts have faced numerous challenges and none have received widespread acceptance.

In the world of parallel computing the Message Passing Interface (MPI) is the de facto standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard Matlab file i/o, resulting in a "pure" Matlab implementation that is exceedingly small (~300 lines of code). Thus, MatlabMPI will run on any combination of computers that Matlab supports. In addition, because of its small size, it is simple to download and use (and modify if you like).

MatlabMPI Page Contents

- [Introduction](#)
- [Download](#)
- [Requirements](#)
- [Installing and Running](#)
- [Launching and File I/O](#)
- [Error Handling](#)
- [Running on Linux](#)
- [Running on MacOSX](#)
- [Running on PC](#)
- [Other Optimizations](#)
- [Running in Batch Mode](#)
- [Other Settings](#)
- [Diagnostics and Troubleshooting](#)
- [First-Time User's Rules of Thumb](#)
- [Files](#)

pMatlab: Parallel Matlab Toolbox

pMatlab provides a set of Matlab data structures and functions that implement distributed Matlab arrays

[to pMatlab page >](#)

uting.iso

Show all downloads...

Add to Matlab path:

```
vdwarf2.ee.bgu.ac.il> cat startup.m  
addpath /usr/local/PP/MatlabMPI/src  
addpath /usr/local/PP/MatlabMPI/examples  
Addpath ./MatMPI
```

xbasic

[illegible]

```
% Initialize MPI.  
MPI_Init;  
  
% Create communicator.  
comm = MPI_COMM_WORLD;  
  
% Modify common directory from default for better performance.  
% comm = MatMPI_Comm_dir(comm, '/tmp');  
  
% Get size and rank.  
comm_size = MPI_Comm_size(comm);  
my_rank = MPI_Comm_rank(comm);  
  
% Print rank.  
disp(['my_rank: ', num2str(my_rank)]);  
  
% Wait momentarily.  
pause(2.0);  
  
% Finalize Matlab MPI.  
MPI_Finalize;  
disp('SUCCESS');  
if (my_rank ~= MatMPI_Host_rank(comm))  
    exit;  
end
```

Demo folder ~/matlab/, watch top at the other machine

```
vdwarf2.ee.bgu.ac.il - PuTTY
vdwarf2.ee.bgu.ac.il> matlab -nodesktop -nodisplay -nojvm

      < M A T L A B >
    Copyright 1984-2007 The MathWorks, Inc.
      Version 7.5.0.338 (R2007b)
      August 9, 2007

-----
Your MATLAB license will expire in 11 days.
Please contact your system administrator or
The MathWorks to renew this license.
-----

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> eval( MPI_Run('xbasic', 2,{'vdwarf3','vdwarf4'}) );
Launching MPI rank: 1 on: vdwarf4
Launching MPI rank: 0 on: vdwarf3

unix_launch =

    rsh vdwarf4 -n 'cd /users/agnon/misc/tel-zur/matlab; /bin/sh ./MatMPI/Unix_Comm
ands.vdwarf4.1.sh &' &
    rsh vdwarf3 -n 'cd /users/agnon/misc/tel-zur/matlab; /bin/sh ./MatMPI/Unix_Comm
ands.vdwarf3.0.sh &' &

>>
>>
>>
>>
>>
>>
>>
>>
```


Parallel Matlab (Octave) using pMatlab

Global arrays – “...Communication is hidden from the programmer; arrays are automatically redistributed when necessary, without the knowledge of the programmer...”

“...The ultimate goal of pMatlab is to move beyond basic messaging (and its inherent programming complexity) towards higher level parallel data structures and functions, allowing any MATLAB user to parallelize their existing program by simply changing and adding a few lines,

Source: http://www.ll.mit.edu/mission/isr/pmatlab/pMatlab_intro.pdf

Instead of:

```
if (my_rank==0) | (my_rank==1) | (my_rank==2) | (my_rank==3)
    A_local=rand(M,N/4);
end

if (my_rank==4) | (my_rank==5) | (my_rank==6) | (my_rank==7)
    B_local=zeros(M/4,N);
end

tag = 0;
if (my_rank==0) | (my_rank==1) | (my_rank==2) | (my_rank==3)
    A_local=fft(A_local);
    for ii = 0:3
        MPI_Send(ii+4, tag, comm, A_local(ii*M/4 + 1:(ii+1)*M/4,:));
    end
end

if (my_rank==4) | (my_rank==5) | (my_rank==6) | (my_rank==7)
    for ii = 0:3
        B_local(:, ii*N/4 + 1:(ii+1)*N/4) = MPI_Recv(ii, tag, comm);
    end
end
```

Write using pMatlab:

```
mapA = map([1 4], {}, [0:3]);  
mapB = map([4 1], {}, [4:7]);  
A = rand(M,N,mapA);  
B = zeros(M,N,mapB);  
B(:, :) = fft(A);
```

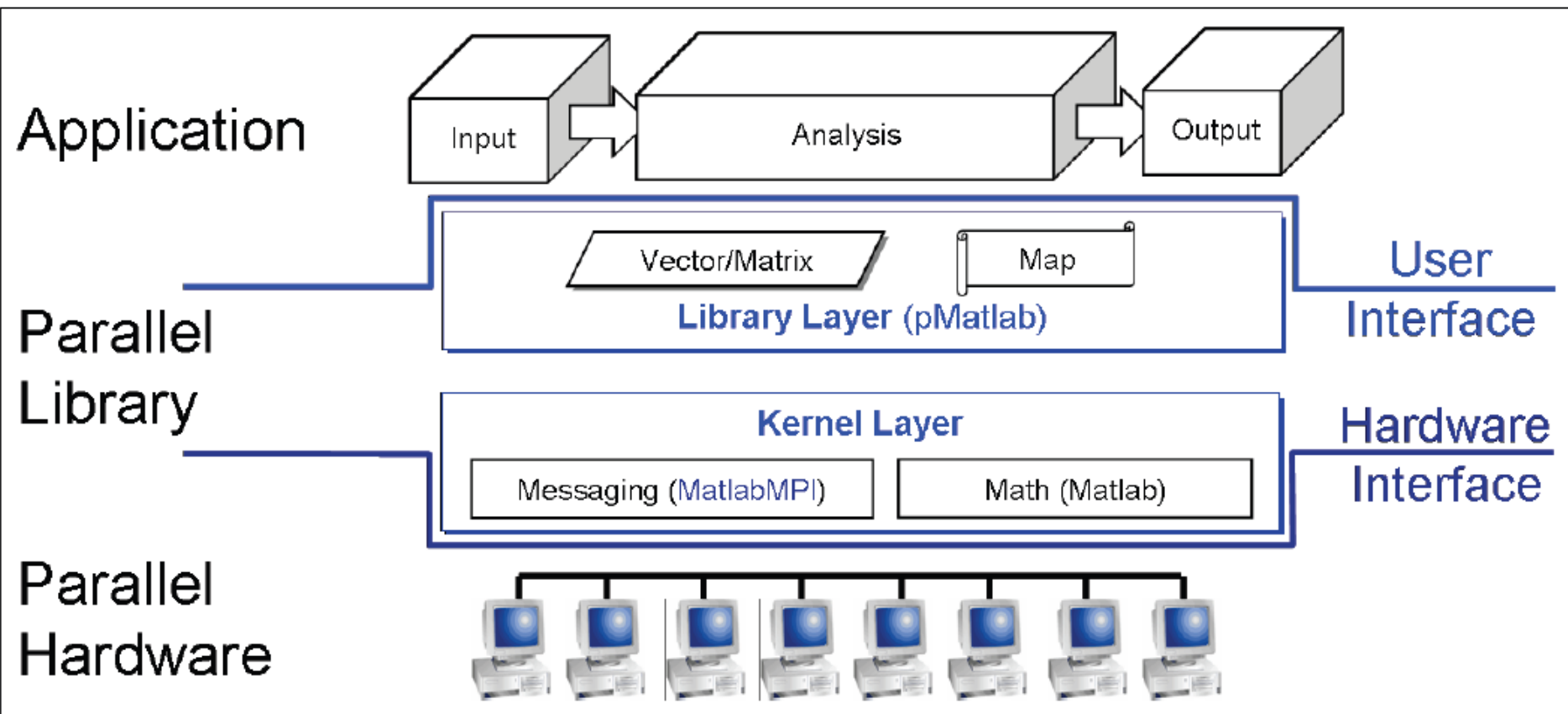


Figure 11 – Parallel MATLAB consists of two layers. pMatlab provides parallel data structures and library functions. MatlabMPI provides messaging capability.

```
-bash-3.1$ matlab -npdisplay
```

```
Warning: Unrecognized MATLAB option "npdisplay".
```

```
MATLAB:118n:InconsistentUiLanguage - The user UI language setting, C, is different from the user locale setting, en_US.UTF-8.
```

```
Warning: No display specified. You will not be able to display graphics on the screen.
```

```
< M A T L A B >
```

```
Copyright 1984-2007 The MathWorks, Inc.
```

```
Version 7.5.0.338 (R2007b)
```

```
August 9, 2007
```

```
To get started, type one of these: helpwin, helpdesk, or demo.
```

```
For product information, visit www.mathworks.com.
```

```
>> eval(pRUN('pHPL',4,{ 'vdwarf1', 'vdwarf2', 'vdwarf3', 'vdwarf4' })))
```

```
Submitting pHPL on 4 processor(s).
```

```
ssh vdwarf1 -n 'kill -9 22302'
```

```
bash: line 0: kill: (22302) - No such process
```

```
ssh vdwarf2 -n 'kill -9 22946'
```

```
bash: line 0: kill: (22946) - No such process
```

```
ssh vdwarf3 -n 'kill -9 4082'
```

```
bash: line 0: kill: (4082) - No such process
```

```
ssh vdwarf4 -n 'kill -9 12431'
```

```
bash: line 0: kill: (12431) - No such process
```

```
Launching MPI rank: 3 on: vdwarf4
```

```
Launching MPI rank: 2 on: vdwarf3
```

```
Launching MPI rank: 1 on: vdwarf2
```

```
Launching MPI rank: 0 on: vdwarf1
```

```
unix_launch =
```

Proceed to pMatlab slides...

Matlab (Octave) + Condor

Sample 1:

```
submit file (cp.sub)
```

```
-----  
universe          = vanilla  
executable        = cp1.bat  
initialdir        = C:\user\CondorMatlab  
log               = matlabtest.log  
error             = matlabtest.err  
input            = CondorMatlabTest.m  
getenv            = true  
requirements      = (NAME == "slot1@remotePC")  
queue
```

cp1.bat

```
-----  
cd "C:\PROGRA~1\MATLAB\R2007b\bin\win32"  
matlab.exe -r "CondorMatlabTest"
```

```
matlab.exe -r "CondorMatlabTest"
```

Condor Demos

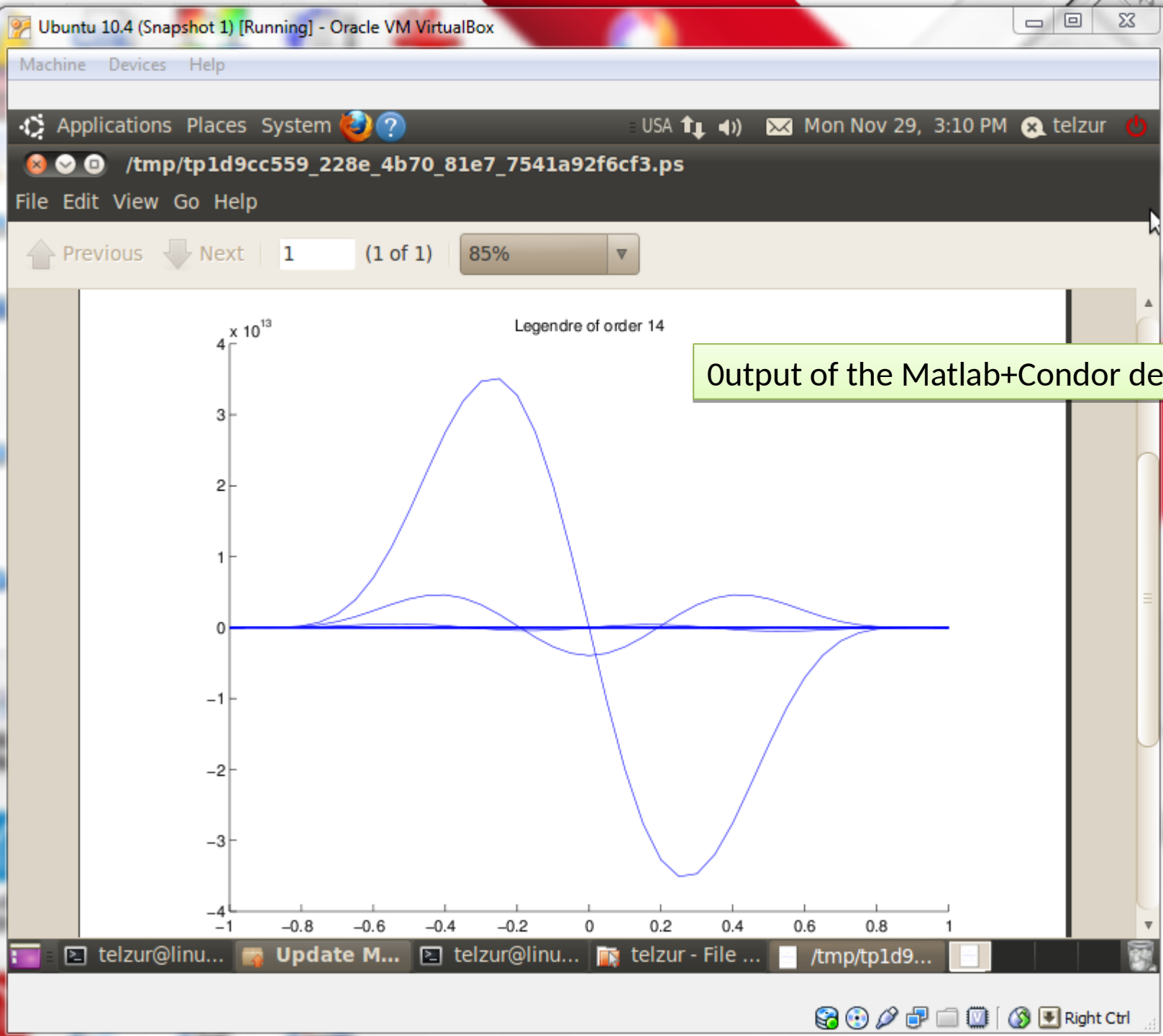
- On my PC: `C:\Users\telzur\Documents\BGU\Teaching\ParallelProcessing\PP2011A\Lectures\06\condor_demo_2010`
- *** has a bug ***

On the Linux vdwarf – Condor + Octave

`/users/agnon/misc/tel-zur/condor/octave`

- On the Linux vdwarf – Condor + Matlab

`/users/agnon/misc/tel-zur/condor/matlab/
example_legendre`



Parallel Matlab(*)

Dr. Guy Tel-Zur

(*)=and other tools

Version: ~~4/11/2018~~, 4/10/2019

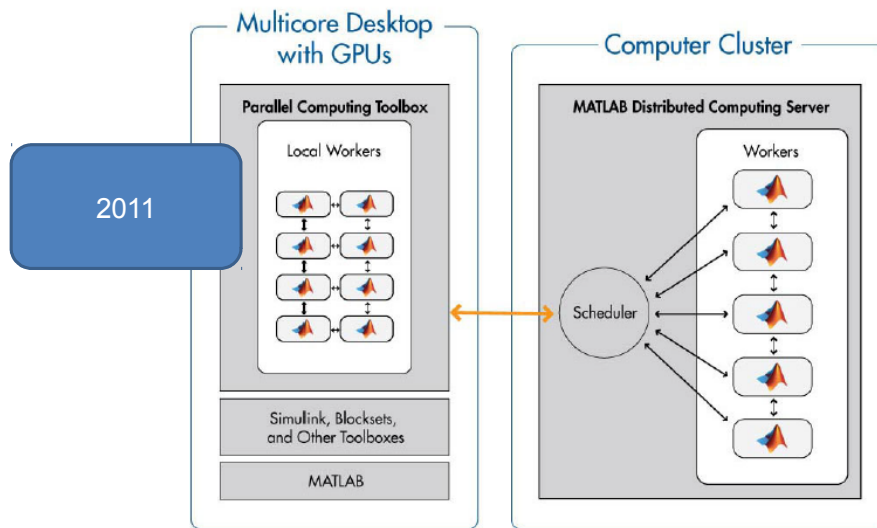
Agenda

- Mathworks – Parallel Computing toolbox
- Parallel Computing with Matlab on Amazon Cloud
- Matlab over GPGPU
- Matlab (Octave) + HTCondor (we will have to learn HTCondor first)
- Parallel Matlab (Octave) using MatlabMPI
- Parallel Matlab (Octave) using pMatlab

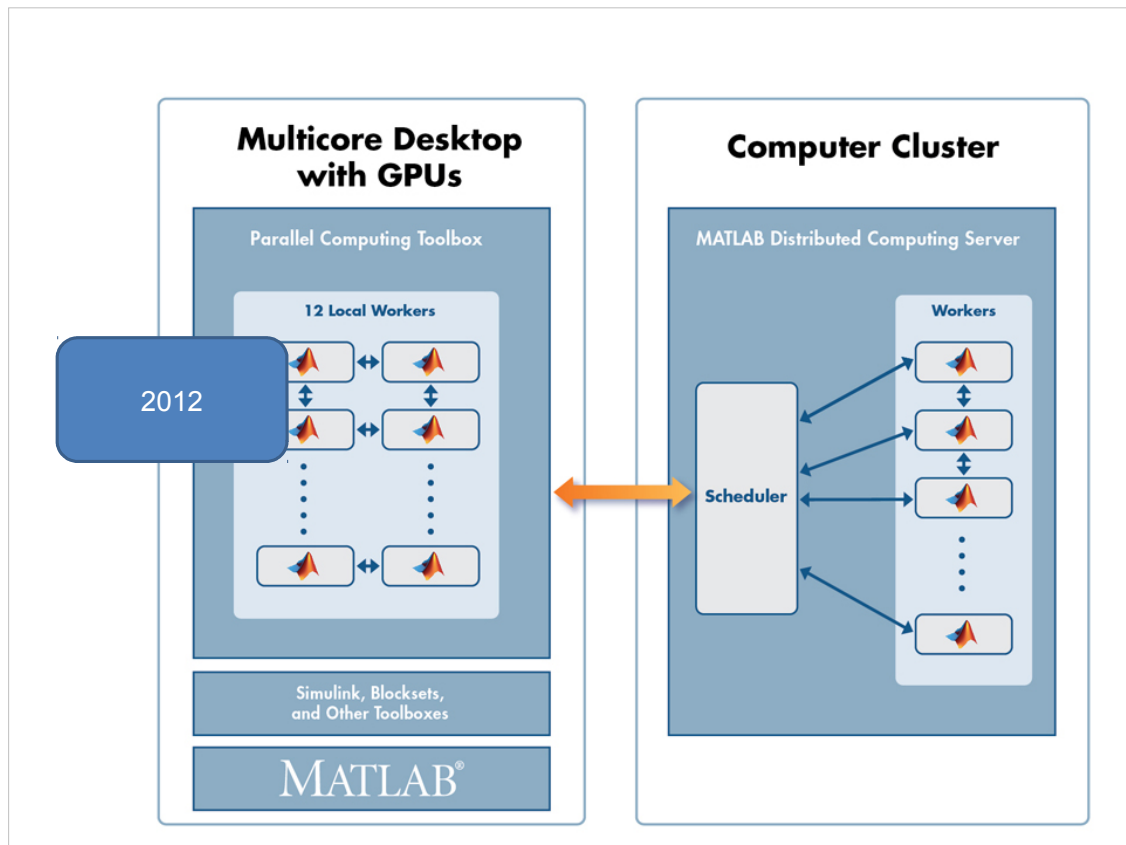
Mathworks – Parallel Computing toolbox

- Parallel Computing without CUDA or MPI(...)
- The toolbox provides “workers” (MATLAB computational engines) to execute applications locally on a multicore desktop
- Parallel for-loops (**parfor**) for running task-parallel algorithms on multiple processors
- Computer cluster and grid support (with MATLAB Distributed Computing Server)

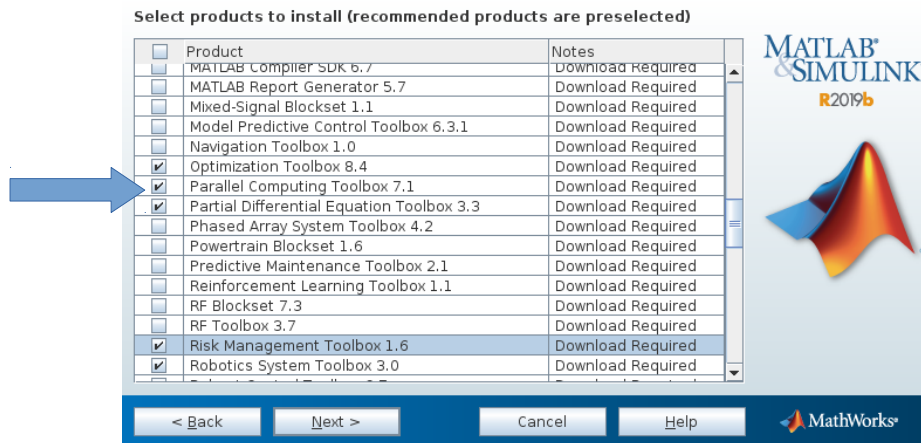
Parallel Computing toolbox



Parallel computing with MATLAB. You can use Parallel Computing Toolbox to run applications on a multicore desktop with eight workers available in the toolbox, take advantage of GPUs, and scale up to a cluster (with MATLAB Distributed Computing Server).



When installing Matlab check the Parallel Computing Toolbox



Local Scheduler Configuration Properties

Configuration name

Description

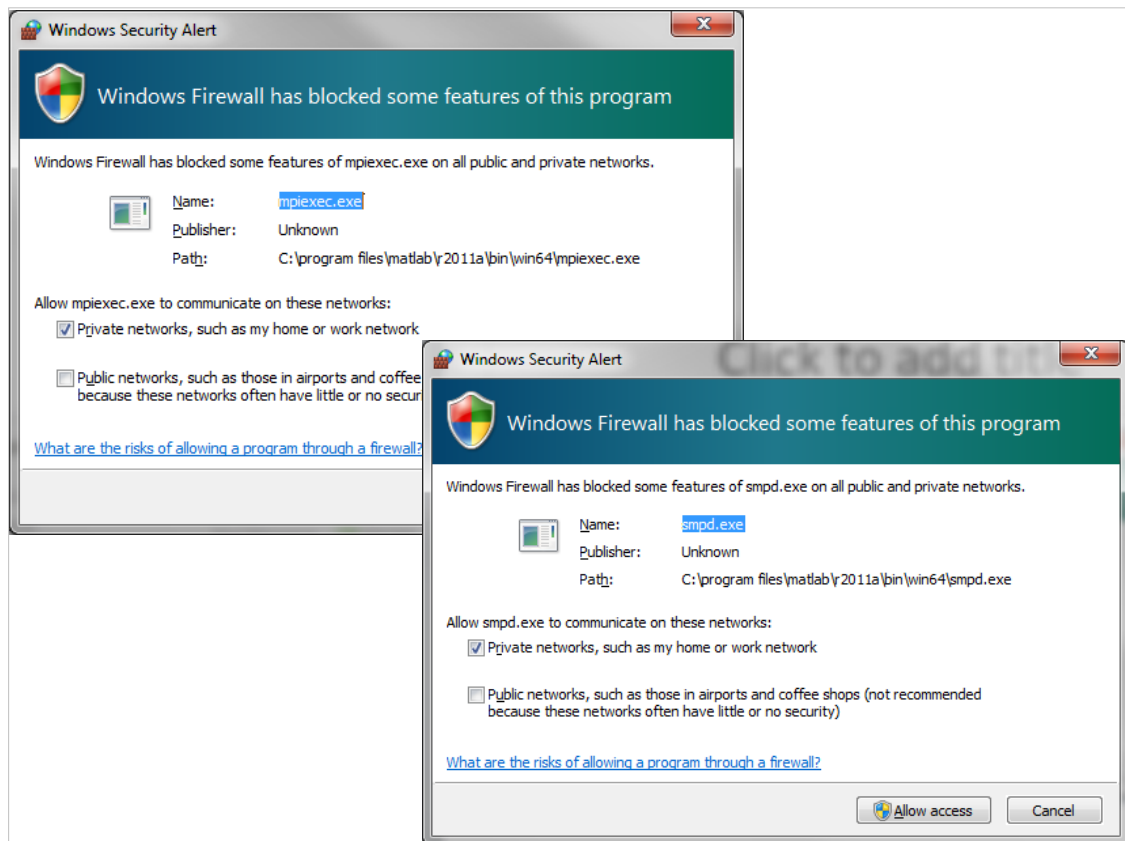
Scheduler Jobs Tasks

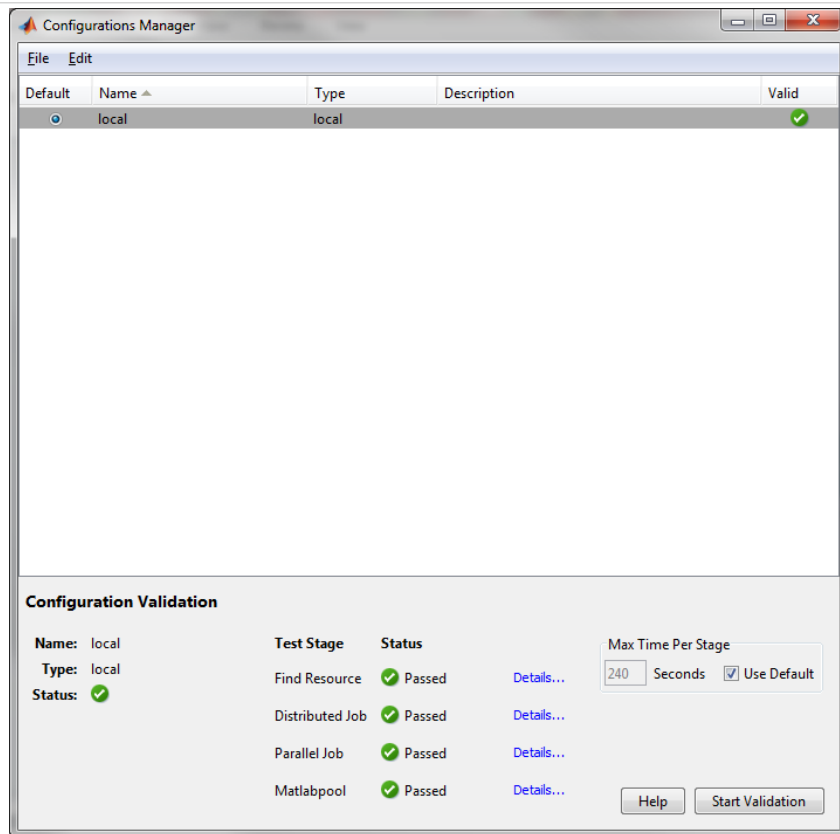
Scheduler type (Type) local

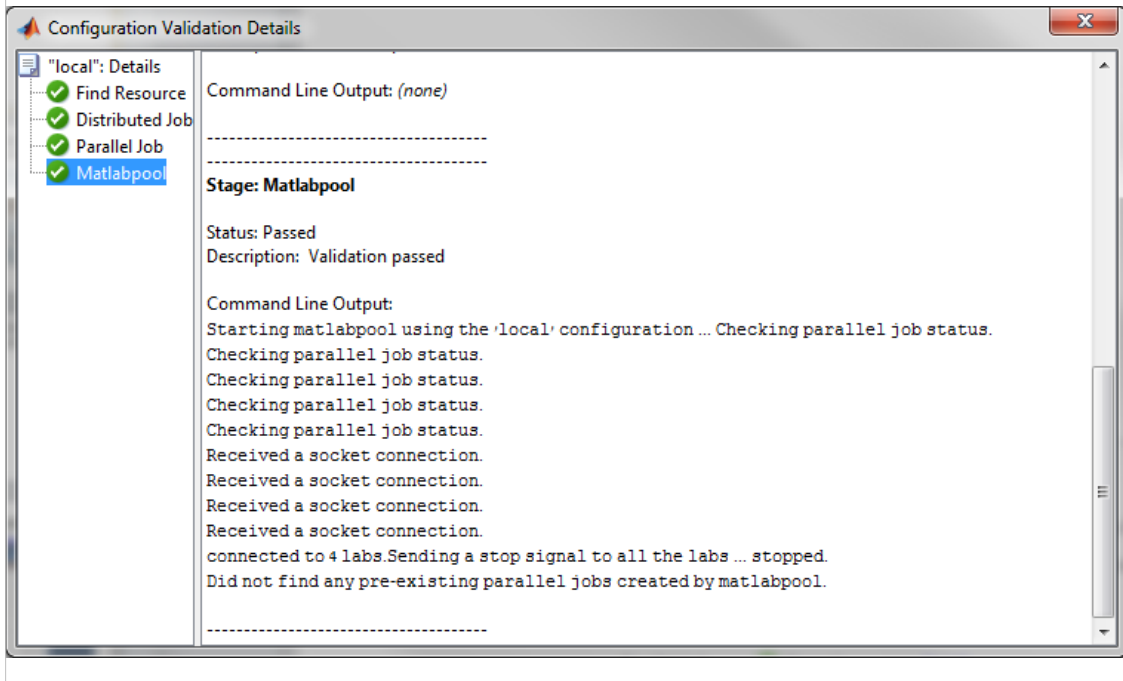
Number of workers available to scheduler (ClusterSize)

Folder where job data is stored (DataLocation)

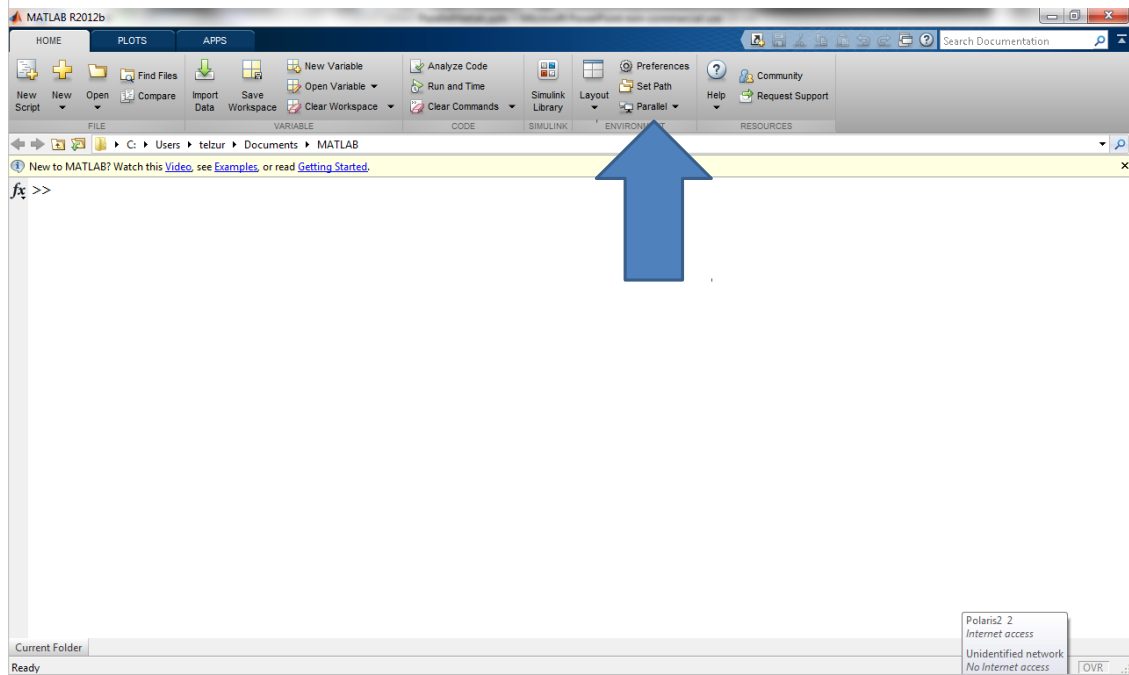
OK Cancel Help







Matlab 2012B



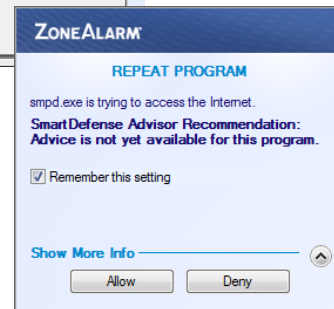
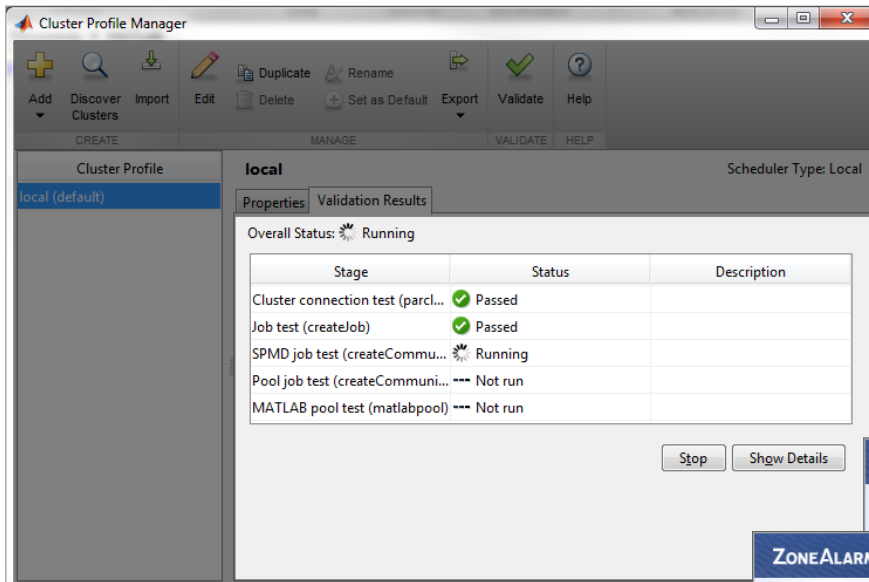
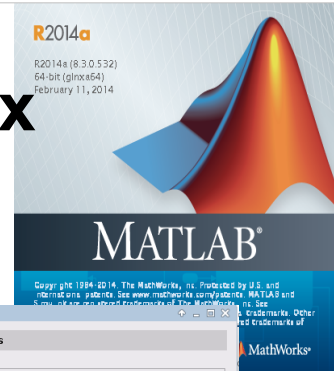
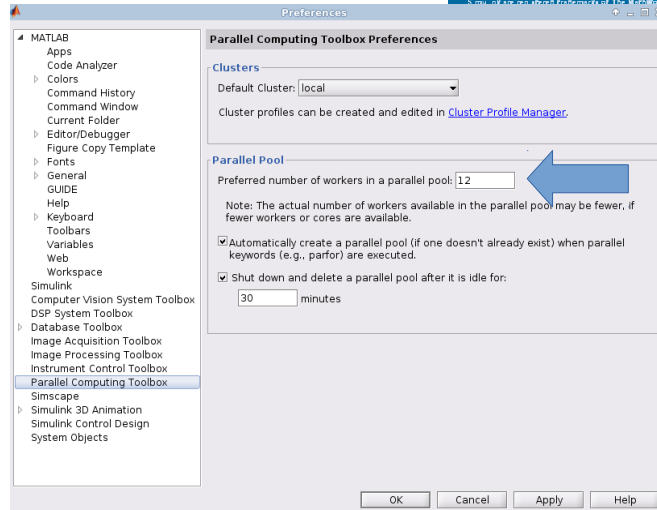
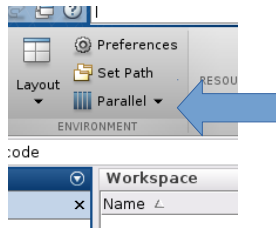


Image Name	User Name	CPU	Memory (P...	Threads	I/O Writes	Image Path Name	Description
System Idle Process	SYSTEM	49	24 K	4			Percentage of time the proc...
MATLAB.exe	telzur	21	158,892 K	28	18	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
MATLAB.exe	telzur	20	159,820 K	28	18	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
MATLAB.exe	telzur	03	302,492 K	45	1,453	C:\Program Files\MATLAB\R2012b\bin\win64\MATLAB.exe	MATLAB (R2012b)
chrome.exe *32	telzur	02	61,620 K	14	32,192	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
svchost.exe	NETWO...	02	8,428 K	28	8,166	C:\Windows\System32\svchost.exe	Host Process for Windows S...
taskmgr.exe	telzur	01	3,428 K	8		C:\Windows\System32\taskmgr.exe	Windows Task Manager
POWERPNT.EXE *32	telzur	00	57,004 K	15	1,389		
chrome.exe *32	telzur	00	6,660 K	14	1,737	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
svchost.exe	LOCAL ...	00	444 K	4		C:\Windows\System32\svchost.exe	Host Process for Windows S...
mpexec.exe	telzur	00	3,572 K	2	106	C:\Program Files\MATLAB\R2012b\bin\win64\mpexec.exe	mpexec.exe
chrome.exe *32	telzur	00	81,452 K	13	16,359	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
spvw64.exe	telzur	00	3,368 K	7	124	C:\Windows\spwov64.exe	Print driver host for 32bit a...
chrome.exe *32	telzur	00	16,304 K	14	33,421	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
chrome.exe *32	telzur	00	20,916 K	13	18,900	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
TosBtHSP.exe *32	telzur	00	960 K	6		C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosBtHSP.exe	TosBtHSP
chrome.exe *32	telzur	00	39,468 K	13	22,828	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
conhost.exe	telzur	00	1,824 K	1		C:\Windows\System32\conhost.exe	Console Window Host
slmssc.exe *32	SYSTEM	00	528 K	7	4	C:\Program Files (x86)\CheckPoint\SSL Network Extender\slmssc.exe	slmssc.exe
chrome.exe *32	telzur	00	5,688 K	13	446	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
avgcsrva.exe	SYSTEM	00	1,548 K	15	585,957	C:\Program Files (x86)\AVG\AVG2013\avgcsrva.exe	AVG Scanning Core Module
TosBtHmg.exe *32	telzur	00	2,088 K	49	167	C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosBtHmg.exe	Bluetooth Manager
TosAIRC.exe *32	telzur	00	720 K	3		C:\Program Files (x86)\TOSHIBA\Bluetooth Toshiba Stack\TosAIRC.exe	TosAIRC
taskeng.exe	SYSTEM	00	2,240 K	6		C:\Windows\System32\taskeng.exe	Task Scheduler Engine
smppd.exe *32	telzur	00	3,544 K	6	7	C:\Program Files\MATLAB\R2012b\bin\win64\smppd.exe	smppd.exe
smppd.exe	telzur	00	9,484 K	13	6,717	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
smppd.exe	telzur	00	4,080 K	8	3	C:\Program Files\MATLAB\R2012b\bin\win64\smppd.exe	smppd.exe
chrome.exe *32	telzur	00	46,536 K	13	5,582	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
chrome.exe *32	telzur	00	5,408 K	13	105	C:\Users\telzur\AppData\Local\Google\Chrome\Application\chrome.exe	Google Chrome
UNS.exe *32	SYSTEM	00	1,064 K	13	2	C:\Program Files (x86)\Intel(R) Management Engine Components\UNS\UNS.exe	User Notification Service

Processes: 135 CPU Usage: 51% Physical Memory: 78%

Version 2014a on Linux

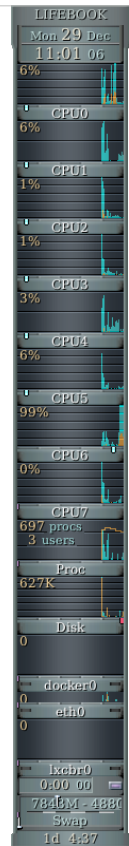


Version 2014a on Linux

The screenshot shows the 'Cluster Profile Manager' application window. The title bar reads 'Cluster Profile Manager'. The menu bar includes 'Add', 'Discover Clusters', 'Import', 'Edit', 'Delete', 'Duplicate', 'Rename', 'Set as Default', 'Export', 'Validate', and 'Help'. The main window is divided into two panes. The left pane, titled 'Cluster Profile', lists 'local (default)'. The right pane, titled 'local', shows 'Type: Local' and 'Properties \ Validation Results \'. The 'Overall Status' is 'Passed' with a green checkmark. Below this is a table with three columns: 'Stage', 'Status', and 'Description'.

Stage	Status	Description
Cluster connection test (parcluster)	Passed	
Job test (createJob)	Passed	
SPMD job test (createCommunicatingJob)	Passed	
Pool job test (createCommunicatingJob)	Passed	
Parallel pool test (parpool)	Passed	

At the bottom right of the right pane are buttons for 'Validate' and 'Show Details'.



parfor - Parallel for loop

parfor - Parallel for loop

Syntax

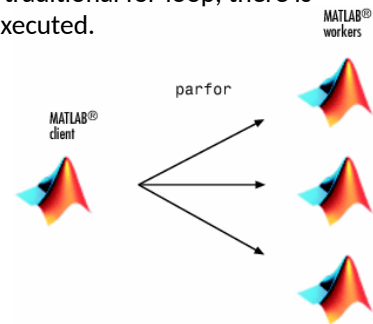
```
parfor loopvar = initval:endval; statements; end  
parfor (loopvar = initval:endval, M); statements; end
```

Description

`parfor loopvar = initval:endval; statements; end` executes a series of MATLAB commands denoted here as *statements* for values of *loopvar* between *initval* and *endval*, inclusive, which specify a vector of increasing integer values. Unlike a traditional for-loop, there is no guarantee of the order in which the loop iterations are executed.

The general format of a `parfor` statement is:

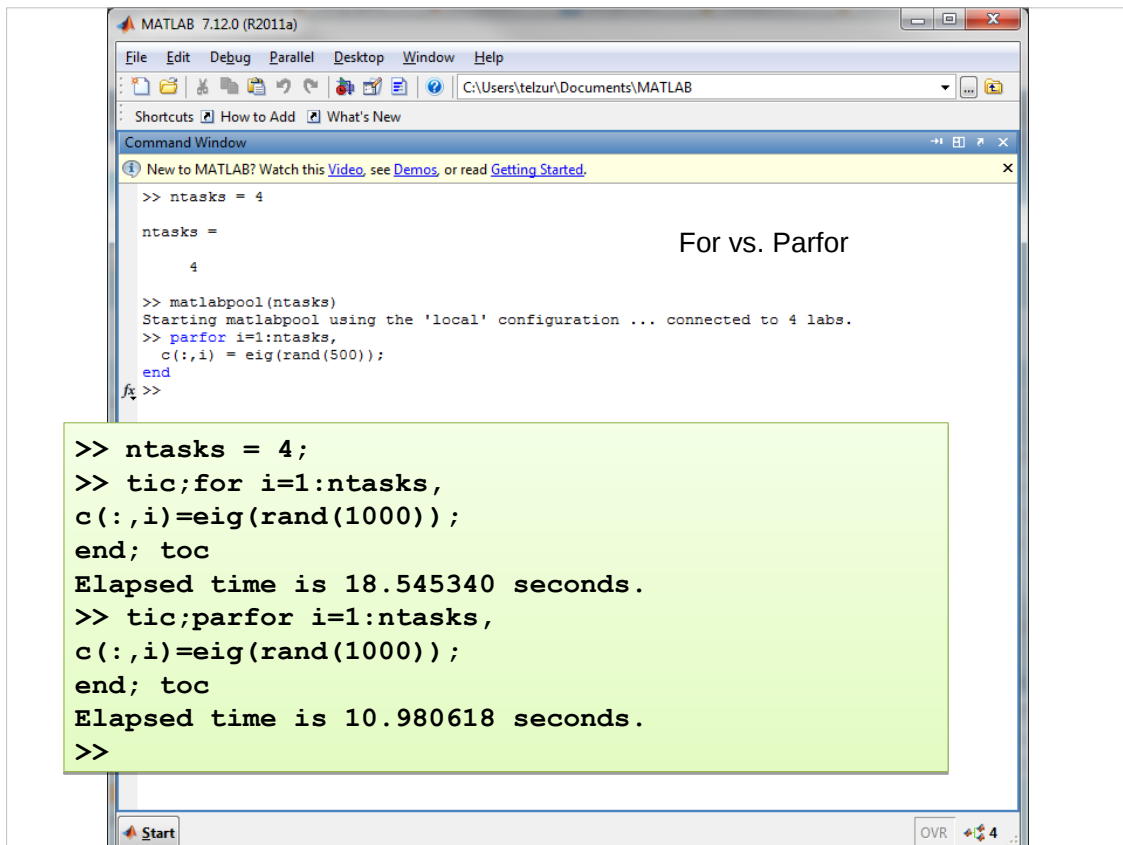
```
parfor loopvar = initval:endval  
    <statements>  
end
```



parfor – an example

Perform three large eigenvalue computations
using three computers or cores:

```
ntasks = 4  
matlabpool(ntasks)  
parfor i=1:ntasks,  
    c(:,i) = eig(rand(500));  
end
```



```

Editor - /home/telzur/Documents/Teaching/BGU/PP/PP2015A/lect
parallel1.m  parallel0.m  +
1 - disp('Serial computation');
2 - ntasks=4;
3 - tic; for i=1:ntasks, c(:,i)=eig(rand(1000));
4 - end; toc
5 -
6 - size(c)
7 -
8 - disp('parallel computation');
9 - delete(gcp);
10 - parpool('local');
11 - tic; parfor i=1:ntasks, d(:,i)=eig(rand(1000));
12 - end; toc
13 - matlabpool('close')
14 -
15 - size(d)

```

Demo: .../lecture09/code/parallel0.m

4 tasks

```

>> parallel0
Serial computation
Elapsed time is 3.374473 seconds.
ans =

    1000         4

parallel computation
Starting parallel pool (parpool) using the 'local' profile ... co
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ... co
Elapsed time is 2.445419 seconds.
Warning: matlabpool will be removed in a future release.
To shutdown a parallel pool use 'delete(gcp('nocreate'))'
instead.
Parallel pool using the 'local' profile is shutting down.
ans =

    1000         4

```

Version 2014a on Linux (i7 processor)

8 tasks

The image shows a MATLAB environment with the following components:

- Editor:** Contains two files: `parallel1.m` and `parallel0.m`. The code in `parallel0.m` is as follows:

```
1 disp('Serial computation');
2 ntasks=8;
3 tic; for i=1:ntasks, c(:,i)=eig(rand(1000));
4 end; toc
5
6 disp('parallel computation');
7 delete(gcf);
8 parpool('local');
9 tic; parfor i=1:ntasks, d(:,i)=eig(rand(1000));
10 end; toc
11
```
- Command Window:** Shows the execution of the `parallel0` script. The output is:

```
>>
>>
>>
>>
>>
>>
>>
>>
>>
>> parallel0
Serial computation
Elapsed time is 7.800062 seconds.
parallel computation
Parallel pool using the 'local' profile is shutting down.
Starting parallel pool (parpool) using the 'local' profile ... conr
Elapsed time is 4.434908 seconds.
>>
```

Two blue arrows point to the elapsed times for the serial and parallel computations.
- Workspace:** A table showing the current workspace variables:

Name	Value
ans	1x1 parallel
c	1000x8 com
d	1000x8 com
i	8
ntasks	8

Demo: ~/lecture09/parallel1.m

```
1 clear all;
2 close all;
3 ntasks = 4;
4 A=rand(2000);
5 disp('Matrix has been generated');
6 % Serial Version:
7 tic
8 for i=1:ntasks
9     B(:,i) = eig(A);
10 end
11 toc
12 disp('done serial')
13 % Parallel Version:
14
15 matlabpool(ntasks)
16 tic
17 parfor i=1:ntasks
18     C(:,i) = eig(A);
19 end
20 toc
21 disp('done parallel')
```

script

Ln 9

Col 20

QVR

Profile Summary

Generated 26-May-2013 21:12:07 using cpu time

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
parallel	1	56.691 s	28.421 s	
parallel_function	1	18.380 s	0.022 s	
parallel_function-distributed_execution	1	18.230 s	0.079 s	
_omp_remoteparfor.getCompleteIntervals	3	18.125 s	0.038 s	
java.util.concurrent.LinkedBlockingQueue (Java method)	20	18.067 s	18.067 s	
matlabpool	1	9.890 s	0.076 s	
MatlabpoolHelper->MatlabpoolHelper.doOpen	1	9.152 s	0.018 s	
_lHelper->MatlabpoolHelper.doMatlabpool	1	9.152 s	0.000 s	
distcomp.interactiveclient.start	1	9.123 s	0.051 s	
distcomp.interactiveclient.pGetSockets	1	8.145 s	0.001 s	
_ient.pGetSockets->iGetSingleConnection	2	8.144 s	7.808 s	
_parseInputsAndCheckOutputsForFunction	1	0.649 s	0.001 s	
_atlabpoolHelper.parseMatlabpoolInputs	1	0.636 s	0.008 s	
_er->ProfileConfigHelper.buildScheduler	1	0.459 s	0.003 s	
parcluster	1	0.456 s	0.080 s	
_lusterAdaptor->iCreateCommunicatingJob	1	0.311 s	0.005 s	
Job.Job->Job.submit	1	0.308 s	-0.000 s	
_gJob->CJSCCommunicatingJob.submitOneJob	1	0.297 s	0.011 s	
Local.hSubmitCommunicatingJob	1	0.280 s	0.013 s	
Local.Local->Local.Local	1	0.209 s	0.018 s	
Cluster.createCommunicatingJob	1	0.184 s	0.002 s	
_hworks.toolbox.distcomp.pmode.Session (Java method)	25	0.174 s	0.174 s	
CJSCluster->CJSCluster.CJSCluster	1	0.169 s	0.048 s	
etime	1036	0.169 s	0.169 s	
CJSSupport->CJSSupport.getProperties	31	0.153 s	0.012 s	
ProfileConfigHelper->iGetDefaultProfile	1	0.149 s	0.000 s	
_er->ProfileConfigHelper.getDefaultName	1	0.149 s	0.000 s	
_Helper->MatlabpoolHelper.checkConfigOk	1	0.149 s	0.000 s	
_oolHelper.checkConfigOk(proffHelper.v)	1	0.149 s	0.000 s	
C:\IS\hM\bin\...\IS\hM\bin\hGetPrivate	74	0.134 s	0.002 s	

parallel1 (1 call, 56.691 sec)

Generated 26-May-2013 21:15:20 using cpu time
script in file C:\Users\teizun\Documents\BGU\Teaching\ParallelProcessing\PP2013\B\lectures\09\parallel1.m
[Copy to new window for comparing multiple runs](#)

This function changed during profiling or before generation of this report. Results may be incomplete or inaccurate.

Refresh

☒ Show parent functions☒ Show busy lines☒ Show child functions
☒ Show Code Analyzer results☒ Show file coverage☒ Show function listing

Parents (calling functions)
No parent

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
9	B(:,i) = eig(A);	2	28.011 s	49.4%	<div></div>
17	parfor i=1:ntasks	1	18.471 s	32.6%	<div></div>
15	matlabpool(ntasks)	1	9.894 s	17.5%	<div></div>
1	clear all;	1	0.195 s	0.3%	
4	A=rand(2000);	1	0.085 s	0.1%	
All other lines			0.035 s	0.1%	
Totals			56.691 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
parallel_function	function	1	18.380 s	32.4%	<div></div>
matlabpool	function	1	9.890 s	17.4%	<div></div>
parfor_endpoint_check	function	2	0 s	0%	
parfor_sliced_fcnhdl_check	function	1	0 s	0%	
close	function	1	0 s	0%	
Selftime (built-ins, overhead, etc.)			28.421 s	50.1%	<div></div>
Totals			56.691 s	100%	

Editor - /home/telzur/Documents/Teaching/BGU/

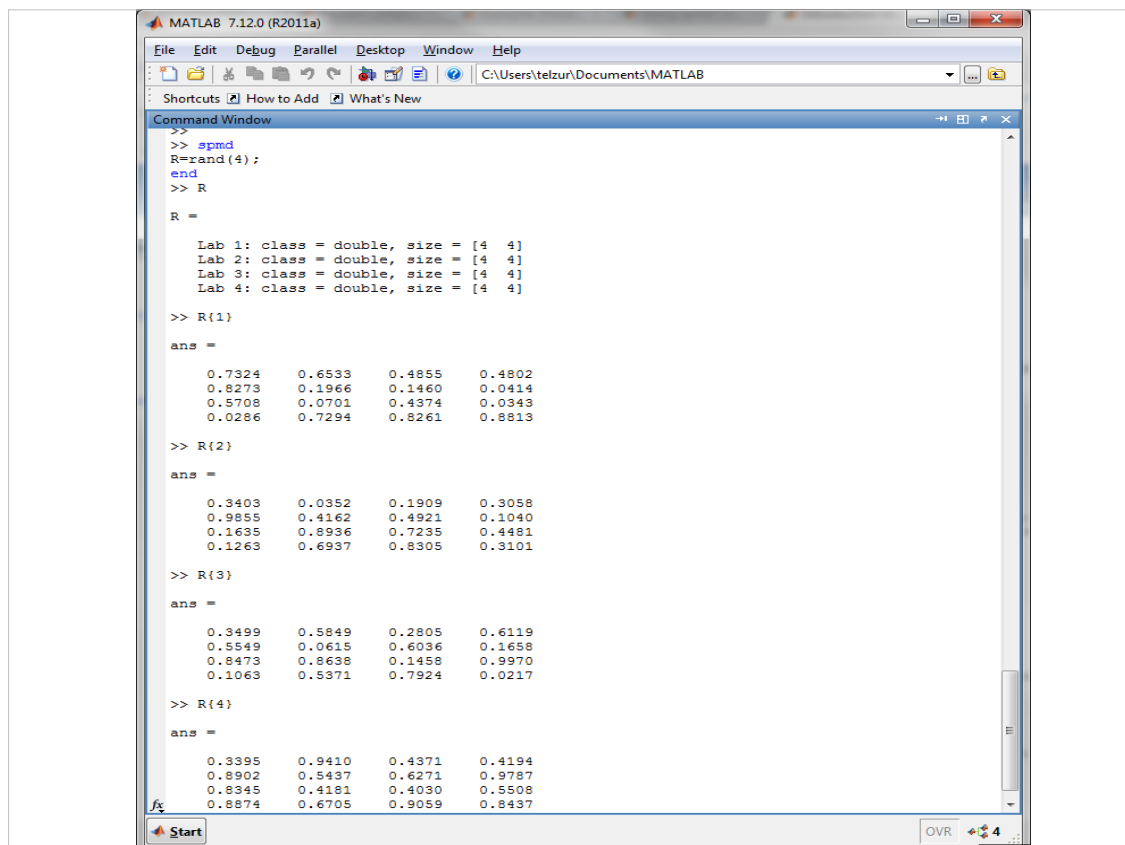
```
parallel.m* x parallel0.m x +
1 - clear all;
2 - close all;
3 - delete(gcf);
4 - ntasks = 8;
5 - A=rand(2000);
6 - disp('Matrix has been generated');
7 - % Serial Version:
8 - tic
9 - for i=1:ntasks
10 -     B(:,i) = eig(A);
11 - end
12 - toc
13 - disp('done serial')
14 - % Parallel Version:
15 - matlabpool('local',ntasks)
16 - tic
17 - parfor i=1:ntasks
18 -     C(:,i) = eig(A);
19 - end
20 - toc
21 - disp('done parallel')
```

Parallel1.m
8 tasks on core i7

```
>> close all
>> clear all
>> parallel
Parallel pool using the 'local' profile is shutting down.
```

```
ntasks =
      8
```

```
Matrix has been generated
Elapsed time is 31.812732 seconds.
done serial
Warning: matlabpool will be removed in a future
release.
Use parpool instead.
Starting matlabpool using the 'local' profile ...
connected to 8 workers.
Elapsed time is 19.150792 seconds.
done parallel
>>
```



optional

←

🏠

Search for add-ons

Q

Installed

Computer Cluster

Tutorials: Parallel and GPU Computing with MATLAB: All in one (9 parts)

version 1.5.0.1 (12.7 KB) by MathWorks Parallel Computing Toolbox Team **STAFF**

Tutorials on Parallel and GPU Computing with MATLAB

★★★★★ 1 Rating

18 Downloads

Updated 1 Sep 2016

[View License](#)

Open Folder

Manage

Overview

Functions

This submission contains all code examples used in tutorial series for Parallel and GPU Computing with MATLAB available here:
<http://www.mathworks.com/products/parallel-computing/tutorials.html>

Topics covered:

1. Product Landscape (no code examples)
2. Prerequisites and Setup (no code examples)
3. Quick Success with parfor
4. Deeper Insights into Using parfor
5. Batch Processing
6. Scaling to Clusters
7. spmd - Parallel Code Beyond parfor
8. Distributed Arrays
9. GPU Computing with MATLAB

Requires

✓ Parallel Computing Toolbox

A NVIDIA CUDA GPU with compute capability 2.0 or above is required for running GPU computing example code

MATLAB Release Compatibility

Created with R2014b

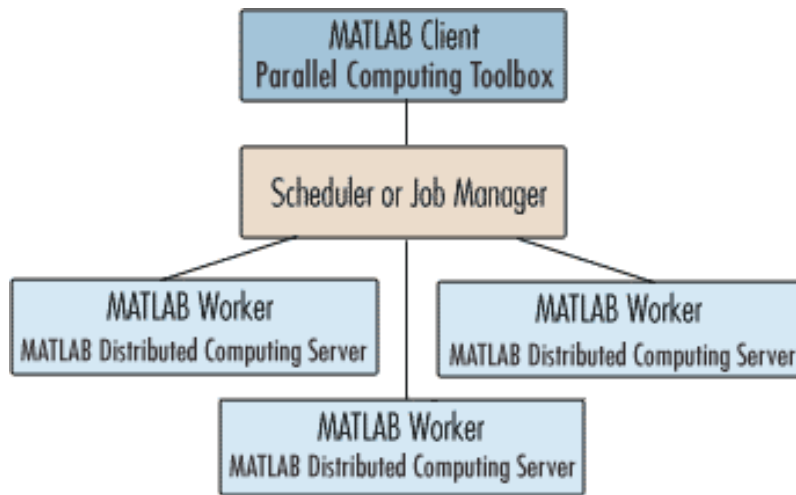
Compatible with any release

Platform Compatibility

☒ Windows ☒ macOS ☒ Linux

<https://www.mathworks.com/videos/series/parallel-and-gpu-computing-tutorials-97719.html>

Parallel Computing Toolbox and MATLAB Distributed Computing



Parallel Computing with Matlab on Amazon Cloud

MATLAB Parallel Computing Tools: Basic Setup and Requirements

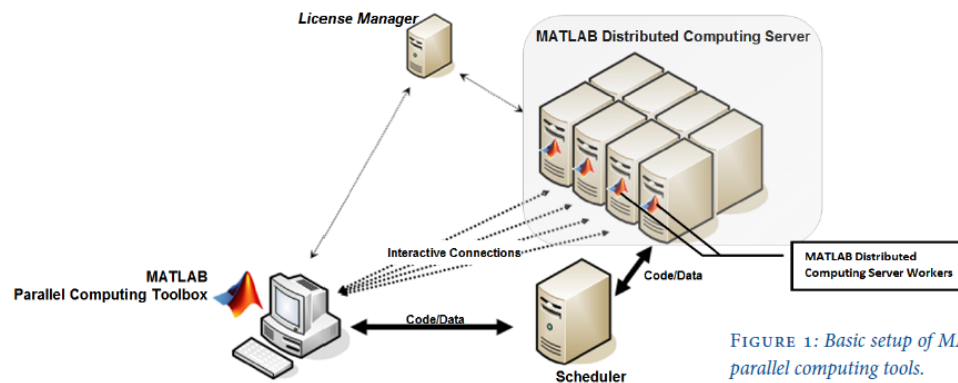


FIGURE 1: Basic setup of MATLAB parallel computing tools.

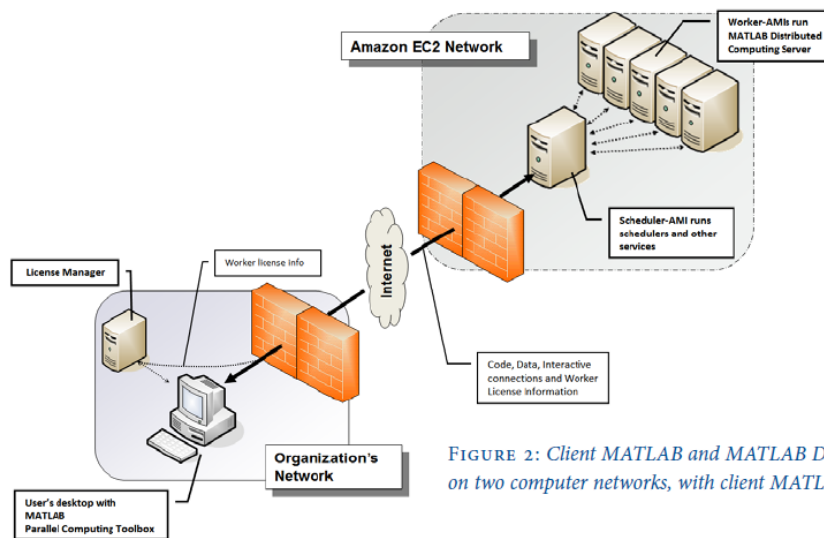
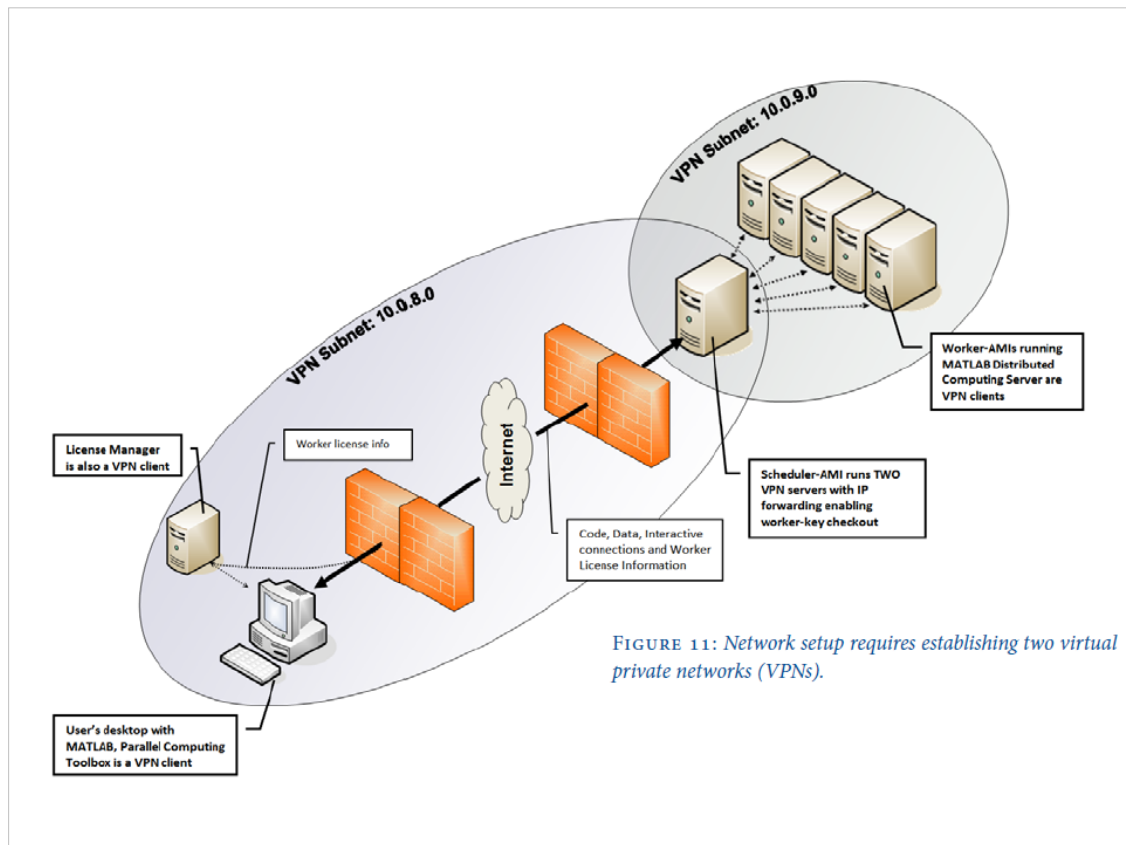
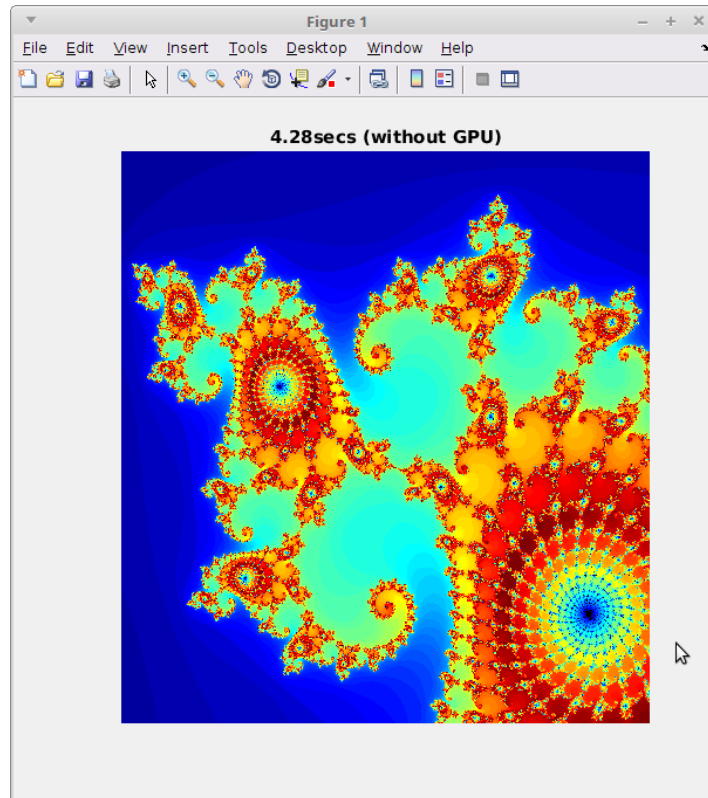


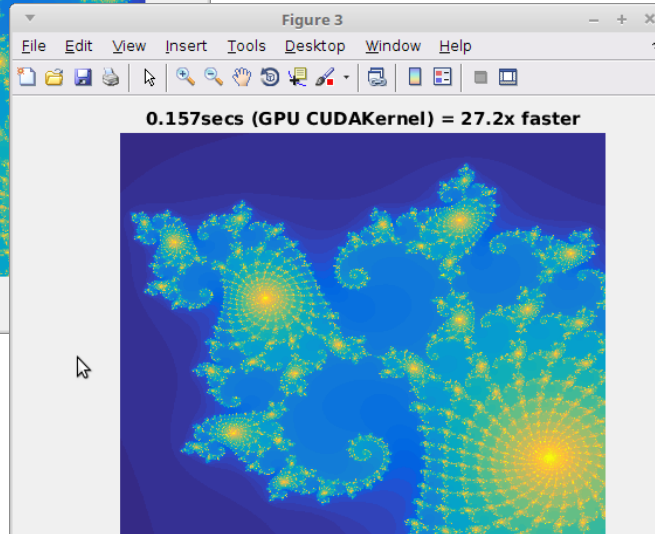
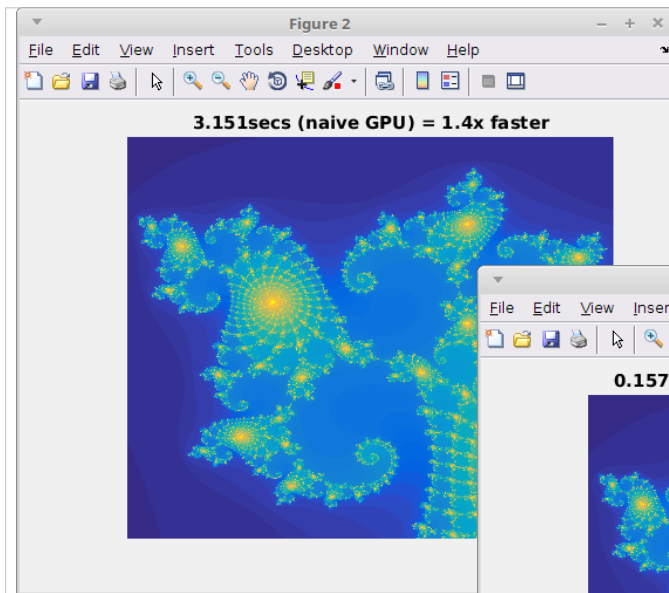
FIGURE 2: Client MATLAB and MATLAB Distributed Computing Server on two computer networks, with client MATLAB on a user's desktop.



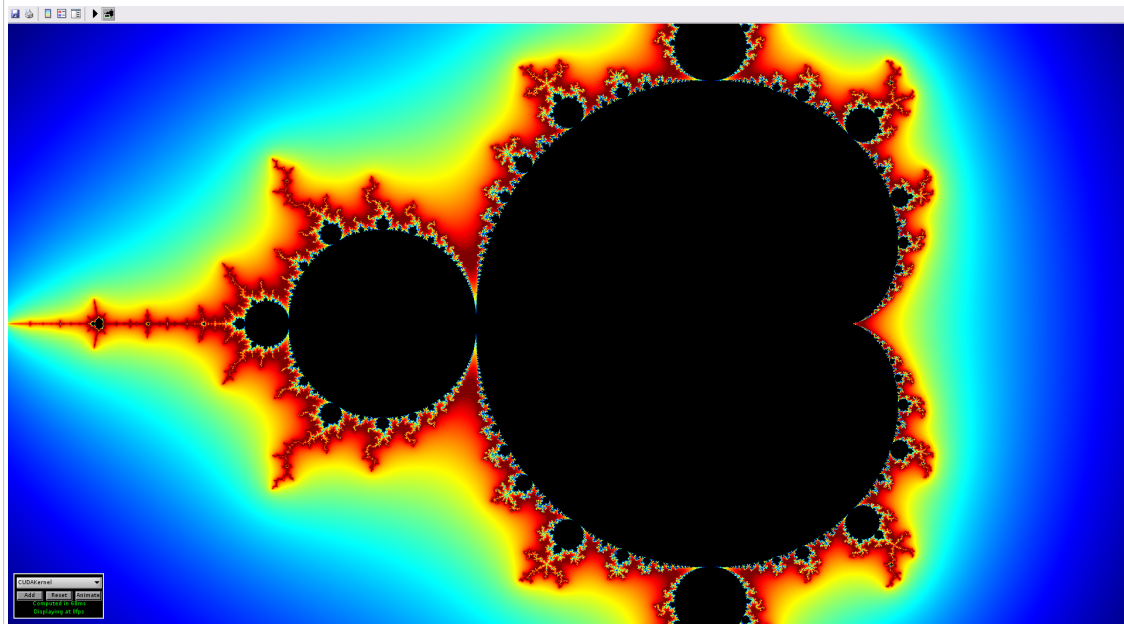
Matlab and GPU computing

`~/.../lectures/08/matlab_code`





MandelbrotViewer



Let's try this

```
A = rand(100, GPUsingle); % A is on GPU memory  
B = rand(100, GPUsingle); % B is on GPU memory  
C = A+B; % executed on GPU.  
D = fft(C); % executed on GPU
```

Executed on GPU

```
A = single(rand(100)); % A is on CPU memory  
B = single(rand(100)); % B is on CPU memory  
C = A+B; % executed on CPU.  
D = fft(C); % executed on CPU
```

Executed on CPU

Matlab Parallel Addons

Contribute

Manage Add-Ons

Clear Filters x Search for add-ons

Filter by Source

☐ MathWorks

5

☐ Community

131

Filter by Category

< Clear Categories

Using MATLAB

Language Fundamentals

879

Data Import and Analysis

998

Mathematics

1,392

Graphics

1,858

Programming

367

App Building

409

Software Development

146

Tools

432

External Language Interfaces

120

Environment and Settings

10

Installation, Licensing, and Activation

10

Parallel Computing

136

Parallel Computing

99

MATLAB Parallel Server

17

Application Deployment

62

136 RESULTS

Parallel Computing (136)



Installed

Parallel Computing Toolbox

Perform parallel computations on multicore computers, GPUs, and clusters

192 Downloads

★★★★★



GPUBench

Compare GPUs using standard numerical benchmarks in MATLAB.

48 Downloads

★★★★★



vfeat/matconvnet

MatConvNet: CNNs for MATLAB

44 Downloads

★★★★★



A GPU Mandelbrot Set

Explore the Mandelbrot Set using MATLAB and a GPU.

44 Downloads

★★★★★



Progress monitor (progress bar)

192 Downloads

★★★★★



Large Data in MATLAB: A Generic Data Processor

48 Downloads

★★★★★



Parallel Computing Toolbox

192 Downloads

★★★★★

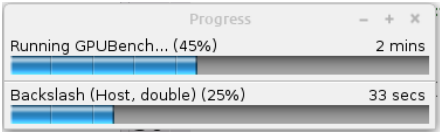


Lynx MATLAB Toolbox

44 Downloads

★★★★★

gpuBench



Processes:						GPU Memory
GPU	PID	Type	Process name			Usage
0	2295	G	/usr/lib/xorg/Xorg			732MiB
0	7182	G	...equest-channel-token=815668599560060867			216MiB
0	21154	C+G	...afa3f6f93/opt/MATLAB/bin/glnxa64/MATLAB			313MiB
0	21831	G	...-token=2ED3348AA2873F9C2157CE208F1F8CA6			1MiB
0	21979	G	...-token=301B7597A0DDCD10126B2777E7CEA836			17MiB

Fri Oct 4 14:30:31 2019

NVIDIA-SMI 430.26						Driver Version: 430.26
GPU	Name	Persistence-M	Bus-Id	Disp.A		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage		
0	GeForce GTX 1050	Off	00000000:01:00:0 Off			N/A
N/A	62C	P0	N/A / N/A	1986MiB / 4042MiB	100%	Default

File Edit View Search Terminal Help										
top - 13:14:28 up 10 days, 19:09, 1 user, load average: 3.86										
Tasks: 424 total, 1 running, 333 sleeping, 0 stopped, 5										
%Cpu0 : 98.7 us, 1.0 sy, 0.0 ni, 0.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu1 : 98.7 us, 1.0 sy, 0.0 ni, 0.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu2 : 5.2 us, 1.4 sy, 0.0 ni, 93.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu3 : 99.0 us, 0.7 sy, 0.0 ni, 0.3 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu4 : 4.8 us, 1.7 sy, 0.0 ni, 93.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu5 : 4.1 us, 3.4 sy, 0.0 ni, 92.1 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu6 : 98.7 us, 1.3 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st										
%Cpu7 : 4.4 us, 2.7 sy, 0.0 ni, 92.6 id, 0.3 wa, 0.0 hi, 0.0 si, 0.0 st										
KiB Mem : 16304712 total, 323108 free, 14073992 used, 1907612 buff/cache										
KiB Swap: 16659452 total, 15949552 free, 709900 used, 12359008 avail Mem										
PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+ COMMAND
21154	telzur	20	0	23.497g	3.097g	191124	S	399.0	19.9	9:20.86 MATLAB
2295	root	20	0	1278804	365348	116224	S	7.9	2.2	71:46.78 Xorg
6325	netdata	20	0	18976	5708	2040	S	3.3	0.0	2:03.39 apps.plugin
25268	telzur	20	0	422408	30760	24132	S	2.6	0.2	0:00.43 mate-screa+
20250	telzur	20	0	533660	30808	12076	S	1.7	0.2	1:38.82 wnck-applet
13327	telzur	20	0	4837236	1.286g	73004	S	1.3	8.3	12:24.06 soffice.bin
20245	telzur	20	0	878104	35008	10948	S	1.3	0.2	8:37.05 marco
2715	netdata	20	0	206856	56076	3032	S	1.0	0.3	57:20.62 netdata
7894	telzur	20	0	2092164	227316	16952	S	1.0	1.4	4:06.71 chromium-b+
23053	telzur	20	0	598156	39100	21508	S	1.0	0.2	1:41.75 mate-termi+

GPU Bench

GPU Comparison Report: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

Summary of results

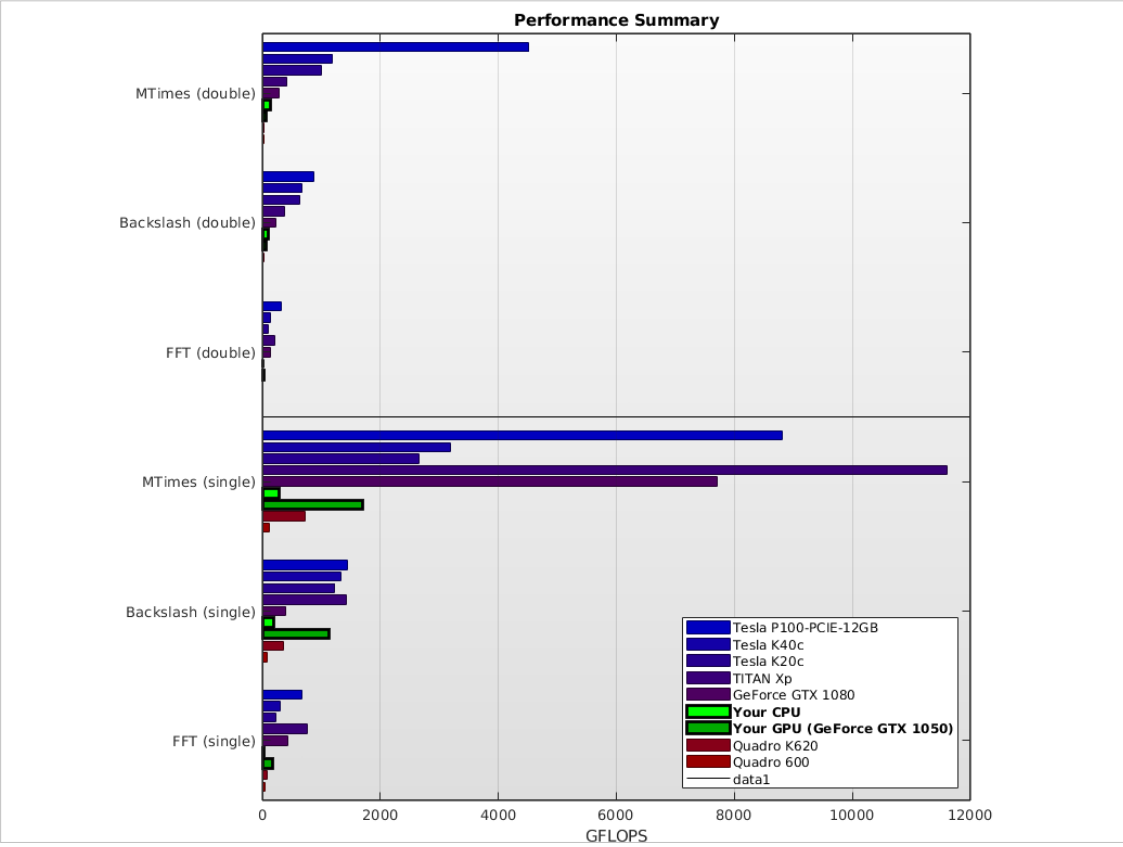
The table and chart below show the peak performance of various GPUs using the same MATLAB version. Your results (if any) are highlighted in bold in the table and on the chart. All other results are from pre-stored data. The peak performance shown is usually achieved when dealing with extremely large arrays. Typical performance in day-to-day use will usually be much lower.

Results captured using the CPUs on the host PC (i.e. without using a GPU) are included for comparison.

Since MATLAB works mostly in double precision the devices are ranked according to how well they perform double-precision calculations. Single precision results are included for completeness. For all results, higher is better.

	Results for data-type 'double' (In GFLOPS)			Results for data-type 'single' (In GFLOPS)		
	MTimes	Backslash	FFT	MTimes	Backslash	FFT
Tesla P100-PCIE-12GB	4518.23	878.97	313.43	8807.20	1439.15	676.20
Tesla K40c	1189.54	677.12	135.88	3187.76	1334.17	294.86
Tesla K20c	1004.06	641.42	106.09	2657.01	1230.28	235.20
TITAN Xp	422.47	371.37	207.24	11607.69	1426.76	763.56
GeForce GTX 1080	280.84	223.05	137.66	7707.01	399.37	424.60
Your CPU	137.45	96.16	14.72	285.93	199.74	21.16
Your GPU (GeForce GTX 1050)	61.24	55.61	31.65	1699.35	1131.12	181.04
Quadro K620	25.45	22.77	12.75	716.71	350.31	75.00
Quadro 600	19.71	17.55	7.62	117.99	88.64	38.58

(click any device name or result to see the detailed data)



GPU results

Results for Backslash (double)

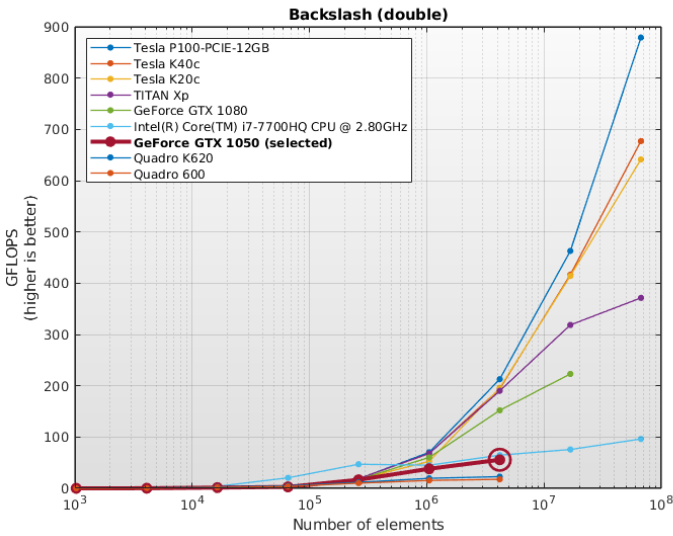
These results show the performance of the GPU or host PC when calculating the [matrix left division](#) of an $N \times N$ matrix with an $N \times 1$ vector. The number of operations is assumed to be $2/3 * N^3 + 3/2 * N^2$.

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Raw data for GeForce GTX 1050 - Backslash (double)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	2.94	0.01
4,096	180,907	1.22	0.15
16,384	1,422,677	1.02	1.40
65,536	11,283,115	4.17	2.71
262,144	89,871,701	5.41	16.62
1,048,576	717,400,747	18.84	38.07
4,194,304	5,732,914,517	103.10	55.61

(N gigafllops = $N \times 10^9$ operations per second)



Results for Backslash (single)

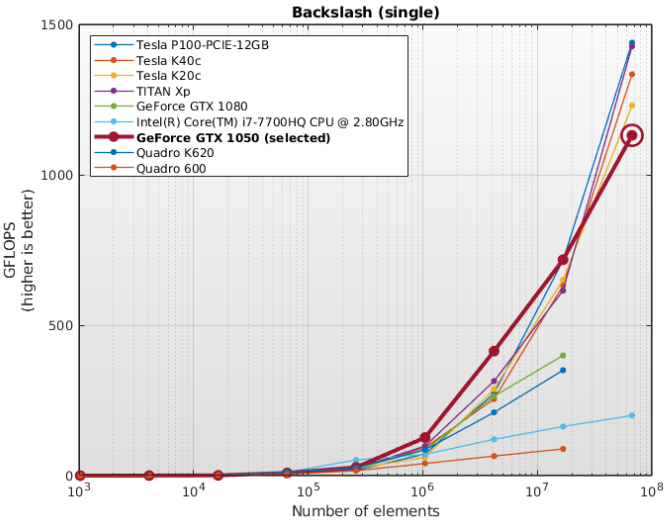
These results show the performance of the GPU or host PC when calculating the [matrix left division](#) of an NxN matrix with an Nx1 vector. The number of operations is assumed to be $\frac{2}{3} * N^3 + \frac{3}{2} * N^2$.

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Raw data for GeForce GTX 1050 - Backslash (single)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	3.47	0.01
4,096	180,907	1.84	0.10
16,384	1,422,677	1.66	0.86
65,536	11,283,115	1.28	8.79
262,144	89,871,701	3.27	27.45
1,048,576	717,400,747	5.67	126.57
4,194,304	5,732,914,517	13.85	413.83
16,777,216	45,838,150,315	63.88	717.55
67,108,864	366,604,539,221	324.11	1131.12

(N gigaflops = $N \times 10^9$ operations per second)



Results for FFT (double)

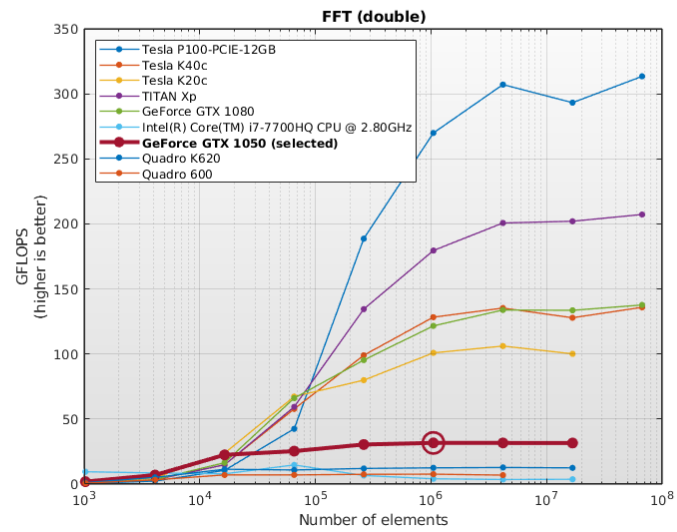
These results show the performance of the GPU or host PC when calculating the **Fast-Fourier-Transform** of a vector of complex numbers. The number of operations for a vector of length N is assumed to be $5 \cdot N \cdot \log_2(N)$.

This calculation is usually memory-bound, i.e. the performance depends mainly on how fast the GPU or host PC can read and write data.

Raw data for GeForce GTX 1050 - FFT (double)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	51,200	0.03	1.87
4,096	245,760	0.04	6.92
16,384	1,146,880	0.05	22.47
65,536	5,242,880	0.21	25.33
262,144	23,592,960	0.78	30.42
1,048,576	104,857,600	3.31	31.65
4,194,304	461,373,440	14.59	31.62
16,777,216	2,013,265,920	63.81	31.55

(N gigaflops = $N \times 10^9$ operations per second)



Results for FFT (single)

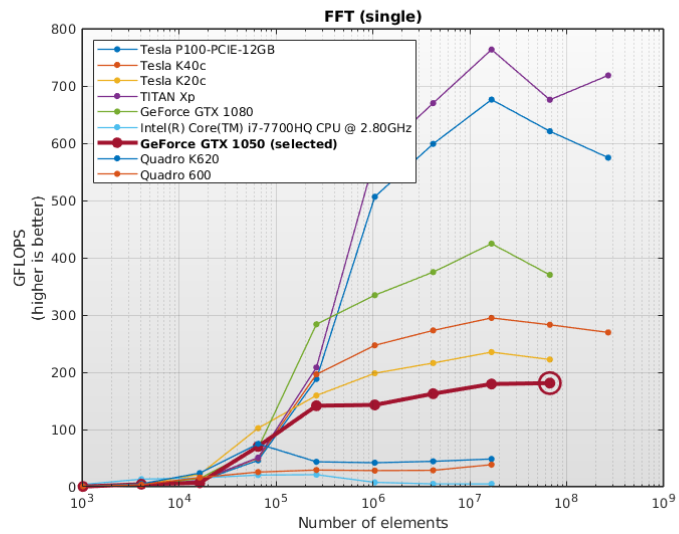
These results show the performance of the GPU or host PC when calculating the [Fast-Fourier-Transform](#) of a vector of complex numbers. The number of operations for a vector of length N is assumed to be $5 \cdot N \cdot \log_2(N)$.

This calculation is usually memory-bound, i.e. the performance depends mainly on how fast the GPU or host PC can read and write data.

Raw data for GeForce GTX 1050 - FFT (single)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	51,200	0.13	0.41
4,096	245,760	0.06	4.44
16,384	1,146,880	0.17	6.90
65,536	5,242,880	0.07	70.38
262,144	23,592,960	0.17	141.56
1,048,576	104,857,600	0.73	142.94
4,194,304	461,373,440	2.84	162.50
16,777,216	2,013,265,920	11.22	179.50
67,108,864	8,724,152,320	48.19	181.04

(N gigaflups = $N \times 10^9$ operations per second)



CPU results

GPU Performance Details: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

- Contents:**
- System Configuration
 - Results for datatype double
 - MTimes (double)
 - Backslash (double)
 - FFT (double)
 - Results for datatype single
 - MTimes (single)
 - Backslash (single)
 - FFT (single)

System Configuration

MATLAB Release: R2019b

Host

Name	Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
Clock	900.349 MHz
Cache	6144 KB
NumProcessors	4
OSType	Linux
OSVersion	buildd@lgw01-amd64-031

Results for Backslash (double)

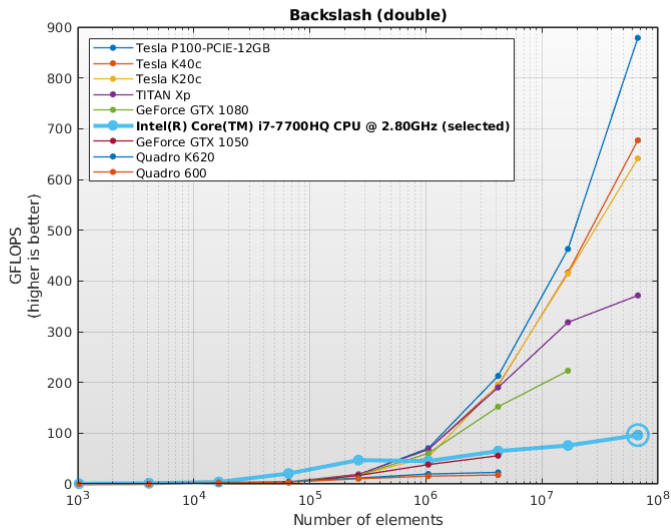
These results show the performance of the GPU or host PC when calculating the matrix left division of an $N \times N$ matrix with an $N \times 1$ vector. The number of operations is assumed to be $\frac{2}{3} * N^3 + \frac{3}{2} * N^2$.

This calculation is usually compute-bound, i.e. the performance depends mainly on how fast the GPU or host PC can perform floating-point operations.

Raw data for Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz - Backslash (double)

Array size (elements)	Num Operations	Time (ms)	GigaFLOPS
1,024	23,381	0.05	0.45
4,096	180,907	0.14	1.30
16,384	1,422,677	0.37	3.81
65,536	11,283,115	0.55	20.47
262,144	89,871,701	1.91	46.94
1,048,576	717,400,747	15.93	45.02
4,194,304	5,732,914,517	88.67	64.65
16,777,216	45,838,150,315	605.55	75.70
67,108,864	366,604,539,221	3812.38	96.16

(N gigaflops = $N \times 10^9$ operations per second)



MatlabMPI and pMatlab

Parallel Matlab (Octave) using MatlabMPI

Files location: vdwarf - /usr/local/PP/MatlabMPI

Read the README there!

cd to the **examples** directory

```
eval( MPI_Run('basic', 3,machines) );  
where:  
machines = {'vdwarf1' 'vdwarf2' 'vdwrf3'}
```


MatlabMPI

<http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html#introduction>

Exit full screen (F11)

 **LINCOLN LABORATORY**
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

[Home](#) [Contact Us](#) [Sitemap](#)

[SEARCH](#)

[About >](#) [Mission Areas >](#) [Employment >](#) [College Recruiting >](#) [News >](#) [Publications >](#) [Outreach >](#) [Workshops/Education >](#)

[Home >](#) [Mission Areas >](#) [ISR Systems and Technology >](#) [MatlabMPI](#)

MATLABMPI

[Space Control >](#)

[Air and Missile Defense Technology >](#)

[Communications and Information Technology >](#)

[ISR Systems and Technology >](#)

- [MatlabMPI](#)
- [pMatlab](#)
- [HPEC Challenge](#)

[Advanced Electronics Technology >](#)

[Tactical Systems >](#)

[Homeland Protection >](#)

[Air Traffic Control >](#)

Parallel Programming with MatlabMPI

Dr. Jeremy Kepner
kepner@ll.mit.edu

I. INTRODUCTION

Matlab is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing and system evaluation. Many of these computations could benefit from faster execution on a parallel computer. There have been many previous attempts to provide an efficient mechanism for running Matlab programs on parallel computers. These efforts have faced numerous challenges and none have received widespread acceptance.

In the world of parallel computing the Message Passing Interface (MPI) is the de facto standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard Matlab file I/O, resulting in a "pure" Matlab implementation that is exceedingly small (<300 lines of code). Thus, MatlabMPI will run on any combination of computers that Matlab supports. In addition, because of its small size, it is simple to download and use (and modify if you like).

MatlabMPI Page Contents

- [Introduction](#)
- [Download](#)
- [Requirements](#)
- [Installing and Running](#)
- [Launching and File I/O](#)
- [Error Handling](#)
- [Running on Linux](#)
- [Running on MacOSX](#)
- [Running on PC](#)
- [Other Optimizations](#)
- [Running in Batch Mode](#)
- [Other Settings](#)
- [Diagnostics and Troubleshooting](#)
- [First-Time User's Rules of Thumb](#)
- [Files](#)

pMatlab: Parallel Matlab Toolbox

pMatlab provides a set of Matlab data structures and functions that implement distributed Matlab arrays

[to pMatlab page >](#)

uting.iso

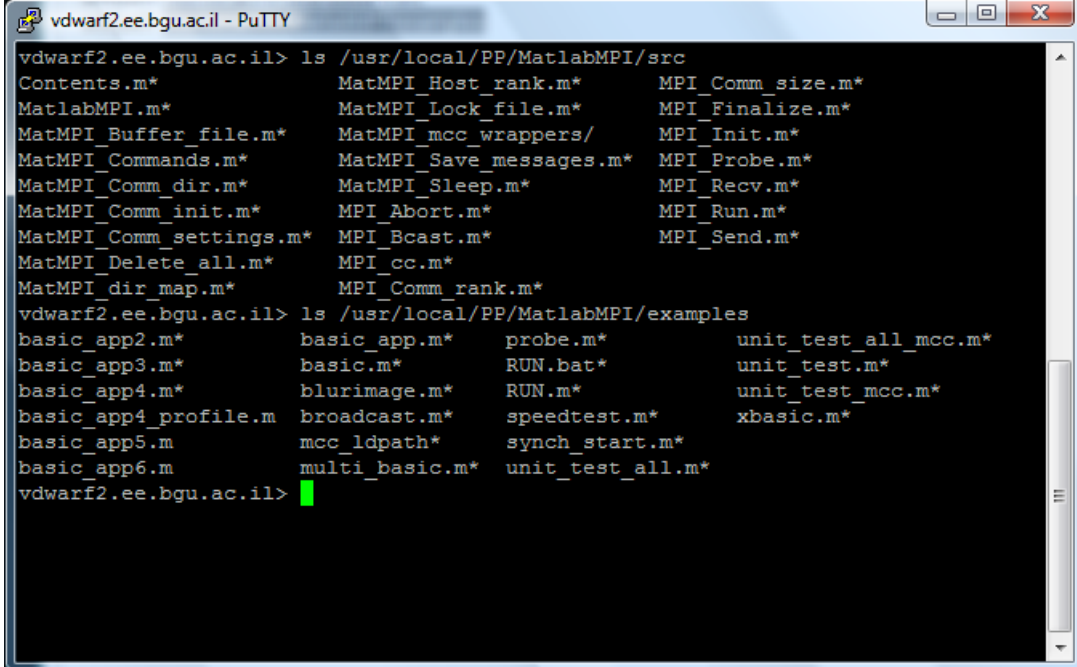
[Show all downloads...](#)

Available examples:

xbasic.m	Extremely simple MatlabMPI program that prints out the rank of each processor.
basic.m	Simple MatlabMPI program that sends data from processor 1 to processor 0.
multi_basic.m	Simple MatlabMPI program that sends data from processor 1 to processor 0 a few times.
probe.m	Simple MatlabMPI program that demonstrates the using MPI_Probe to check for incoming messages.
broadcast.m	Tests MatlabMPI broadcast command.
basic_app.m	Examples of the most common usages of MatlabMPI.
basic_app2.m	Examples of the most common usages of MatlabMPI.
basic_app3.m	Examples of the most common usages of MatlabMPI.
basic_app4.m	Examples of the most common usages of MatlabMPI.
blurimage.m	MatlabMPI test parallel image processing application.
speedtest.m	Times MatlabMPI for a variety of messages.
synch_start.m	Function for synchronizing starts.
machines.m	Example script for creating a machine description.
unit_test.m	Wrapper for using an example as a unit test.
unit_test_all.m	Calls all of the examples as way of testing the entire library.
unit_test_mcc.m	Wrapper for using an example as a mcc unit test.
unit_test_all_mcc.m	Calls all of the examples using MPI_cc as way of testing the entire library.

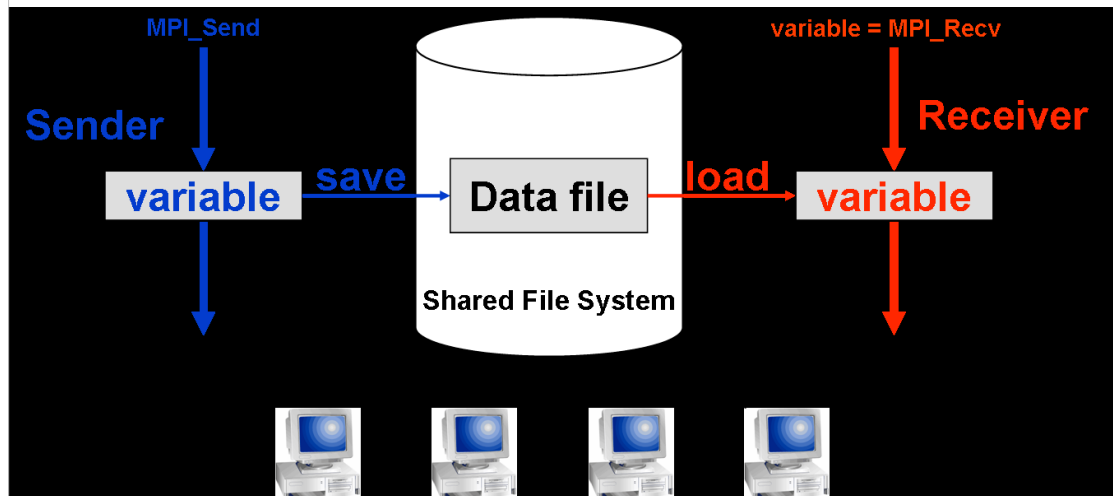
MatlabMPI Demo

Installed on the vdwarf machines



```
vdwarf2.ee.bgu.ac.il - PuTTY
vdwarf2.ee.bgu.ac.il> ls /usr/local/PP/MatlabMPI/src
Contents.m*           MatMPI_Host_rank.m*   MPI_Comm_size.m*
MatlabMPI.m*          MatMPI_Lock_file.m*   MPI_Finalize.m*
MatMPI_Buffer_file.m* MatMPI_mcc_wrappers/  MPI_Init.m*
MatMPI_Commands.m*    MatMPI_Save_messages.m* MPI_Probe.m*
MatMPI_Comm_dir.m*    MatMPI_Sleep.m*       MPI_Recv.m*
MatMPI_Comm_init.m*   MPI_Abort.m*          MPI_Run.m*
MatMPI_Comm_settings.m* MPI_Bcast.m*          MPI_Send.m*
MatMPI_Delete_all.m*  MPI_cc.m*
MatMPI_dir_map.m*     MPI_Comm_rank.m*
vdwarf2.ee.bgu.ac.il> ls /usr/local/PP/MatlabMPI/examples
basic_app2.m*      basic_app.m*      probe.m*      unit_test_all_mcc.m*
basic_app3.m*      basic.m*          RUN.bat*      unit_test.m*
basic_app4.m*      blurimage.m*      RUN.m*        unit_test_mcc.m*
basic_app4_profile.m broadcast.m*      speedtest.m*  xbasic.m*
basic_app5.m       mcc_ldpath*      synch_start.m*
basic_app6.m       multi_basic.m*   unit_test_all.m*
vdwarf2.ee.bgu.ac.il>
```

MatlabMPI implements the fundamental communication operations in MPI using MATLAB's file I/O functions.



MatlabMPI

<http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html#introduction>

Exit full screen (F11)

LINCOLN LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Home

Contact Us

Sitemap

SEARCH

About >

Mission Areas >

Employment >

College Recruiting >

News >

Publications >

Outreach >

Workshops/Education >

Home > Mission Areas > ISR Systems and Technology > MatlabMPI

Space Control >

Air and Missile Defense Technology >

Communications and Information Technology >

ISR Systems and Technology >

MatlabMPI

pMatlab

HPEC Challenge

Advanced Electronics Technology >

Tactical Systems >

Homeland Protection >

Air Traffic Control >

MATLABMPI

Parallel Programming with MatlabMPI

Dr. Jeremy Kepner

kepner@ll.mit.edu

I. INTRODUCTION

Matlab is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing and system evaluation. Many of these computations could benefit from faster execution on a parallel computer. There have been many previous attempts to provide an efficient mechanism for running Matlab programs on parallel computers. These efforts have faced numerous challenges and none have received widespread acceptance.

In the world of parallel computing the Message Passing Interface (MPI) is the de facto standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard Matlab file I/O, resulting in a "pure" Matlab implementation that is exceedingly small (<300 lines of code). Thus, MatlabMPI will run on any combination of computers that Matlab supports. In addition, because of its small size, it is simple to download and use (and modify if you like).

MatlabMPI Page Contents

Introduction

Download

Requirements

Installing and Running

Launching and File I/O

Error Handling

Running on Linux

Running on MacOSX

Running on PC

Other Optimizations

Running in Batch Mode

Other Settings

Diagnostics and Troubleshooting

First-Time User's Rules of Thumb

Files

pMatlab: Parallel Matlab Toolbox

pMatlab provides a set of Matlab data structures and functions that implement distributed Matlab arrays

to pMatlab page >

uting.iso

Show all downloads...

Add to Matlab path:

```
vdwarf2.ee.bgu.ac.il> cat startup.m  
addpath /usr/local/PP/MatlabMPI/src  
addpath /usr/local/PP/MatlabMPI/examples  
Addpath ./MatMPI
```

xbasic

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Basic Matlab MPI script that
% prints out a rank.
%
% To run, start Matlab and type:
%
%   eval( MPI_Run('xbasic',2,{}) );
%
% Or, to run a different machine type:
%
%   eval( MPI_Run('xbasic',2,{'machine1' 'machine2'}) );
%
% Output will be piped into two files:
%
%   MatMPI/xbasic.0.out
%   MatMPI/xbasic.1.out
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MatlabMPI
% Dr. Jeremy Kepner
% MIT Lincoln Laboratory
% kepner@ll.mit.edu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Initialize MPI.
MPI_Init;

% Create communicator.
comm = MPI_COMM_WORLD;

% Modify common directory from default for better performance.
% comm = MatMPI_Comm_dir(comm, '/tmp');

% Get size and rank.
comm_size = MPI_Comm_size(comm);
my_rank = MPI_Comm_rank(comm);

% Print rank.
disp(['my_rank: ', num2str(my_rank)]);

% Wait momentarily.
pause(2.0);

% Finalize Matlab MPI.
MPI_Finalize;
disp('SUCCESS');
if (my_rank ~= MatMPI_Host_rank(comm))
    exit;
end
```


Demo folder ~/matlab/, watch top at the other machine

```
vdwarf2.ee.bgu.ac.il - PuTTY
vdwarf2.ee.bgu.ac.il> matlab -nodesktop -nodisplay -nojvm

      < M A T L A B >
    Copyright 1984-2007 The MathWorks, Inc.
      Version 7.5.0.338 (R2007b)
      August 9, 2007

-----
Your MATLAB license will expire in 11 days.
Please contact your system administrator or
The MathWorks to renew this license.
-----

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> eval( MPI_Run('xbasic', 2,{'vdwarf3','vdwarf4'}) );
Launching MPI rank: 1 on: vdwarf4
Launching MPI rank: 0 on: vdwarf3

unix_launch =

    rsh vdwarf4 -n 'cd /users/agnon/misc/tel-zur/matlab; /bin/sh ./MatMPI/Unix_Comm
ands.vdwarf4.1.sh &' &
    rsh vdwarf3 -n 'cd /users/agnon/misc/tel-zur/matlab; /bin/sh ./MatMPI/Unix_Comm
ands.vdwarf3.0.sh &' &

>>
>>
>>
>>
>>
>>
>>
>>
```

Parallel Matlab (Octave) using pMatlab

Global arrays – “...Communication is hidden from the programmer; arrays are automatically redistributed when necessary, without the knowledge of the programmer...”

“...The ultimate goal of pMatlab is to move beyond basic messaging (and its inherent programming complexity) towards higher level parallel data structures and functions, allowing any MATLAB user to parallelize their existing program by simply changing and adding a few lines,

Source: http://www.ll.mit.edu/mission/isr/pmatlab/pMatlab_intro.pdf

Instead of:

```
if (my_rank==0) | (my_rank==1) | (my_rank==2) | (my_rank==3)
    A_local=rand(M,N/4);
end

if (my_rank==4) | (my_rank==5) | (my_rank==6) | (my_rank==7)
    B_local=zeros(M/4,N);
end

tag = 0;
if (my_rank==0) | (my_rank==1) | (my_rank==2) | (my_rank==3)
    A_local=fft(A_local);
    for ii = 0:3
        MPI_Send(ii+4, tag, comm, A_local(ii*M/4 + 1:(ii+1)*M/4,:));
    end
end

if (my_rank==4) | (my_rank==5) | (my_rank==6) | (my_rank==7)
    for ii = 0:3
        B_local(:, ii*N/4 + 1:(ii+1)*N/4) = MPI_Recv(ii, tag, comm);
    end
end
```

Write using pMatlab:

```
mapA = map([1 4], {}, [0:3]);  
mapB = map([4 1], {}, [4:7]);  
A = rand(M,N,mapA);  
B = zeros(M,N,mapB);  
B(:, :) = fft(A);
```

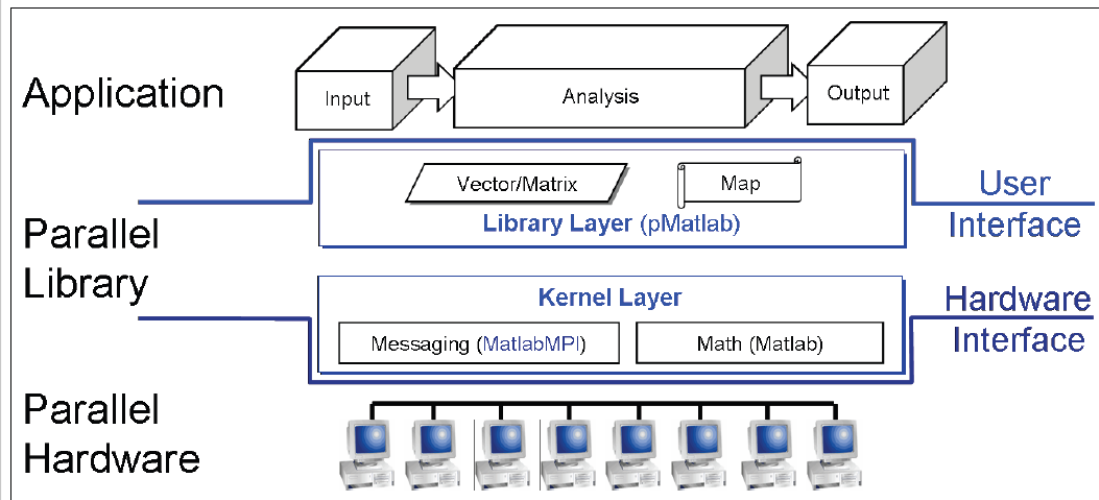


Figure 11 – Parallel MATLAB consists of two layers. pMatlab provides parallel data structures and library functions. MatlabMPI provides messaging capability.

```
eessrv.ee.bgu.ac.il - PuTTY
-bash-3.1$ matlab -npdisplay
Warning: Unrecognized MATLAB option "npdisplay".
MATLAB:118n:InconsistentUILanguage - The user UI language setting, C, is different from the user locale setting, en_US.UTF-8.
Warning: No display specified. You will not be able to display graphics on the screen.

< M A T L A B >
Copyright 1984-2007 The MathWorks, Inc.
Version 7.5.0.338 (R2007b)
August 9, 2007

To get started, type one of these: helpwin, helpdesk, or demo.
For product information, visit www.mathworks.com.

>> eval(pRUN('pHPL',4,{'vdwarf1','vdwarf2','vdwarf3','vdwarf4'}))
Submitting pHPL on 4 processor(s).
ssh vdwarf1 -n 'kill -9 22302'
bash: line 0: kill: (22302) - No such process
ssh vdwarf2 -n 'kill -9 22946'
bash: line 0: kill: (22946) - No such process
ssh vdwarf3 -n 'kill -9 4082'
bash: line 0: kill: (4082) - No such process
ssh vdwarf4 -n 'kill -9 12431'
bash: line 0: kill: (12431) - No such process
Launching MPI rank: 3 on: vdwarf4
Launching MPI rank: 2 on: vdwarf3
Launching MPI rank: 1 on: vdwarf2
Launching MPI rank: 0 on: vdwarf1

unix_launch =
```

Proceed to pMatlab slides...

Matlab (Octave) + Condor

Sample 1:

```
submit file (cp.sub)
-----
universe          = vanilla
executable        = cp1.bat
initialdir        = C:\user\CondorMatlab
log               = matlabtest.log
error             = matlabtest.err
input             = CondorMatlabTest.m
getenv            = true
requirements      = (NAME == "slot1@remotePC")
queue
-----
```

cp1.bat

```
-----
cd "C:\PROGRA~1\MATLAB\R2007b\bin\win32"
matlab.exe -r "CondorMatlabTest"
```

matlabtest.log . CondorMatlabTest

Condor Demos

- On my PC: C:\Users\telzur\Documents\BGU\Teaching\ParallelProcessing\PP2011A\Lectures\06\condor_demo_2010
- *** has a bug ***

On the Linux vdwarf – Condor + Octave

/users/agnon/misc/tel-zur/condor/octave

- On the Linux vdwarf – Condor + Matlab

/users/agnon/misc/tel-zur/condor/matlab/
example_legendre

