Introduction to the MATLAB Parallel Toolbox

Marinus Pennings

October 10,2017

DIVISION OF RESEARCH
TEXAS A&M UNIVERSITY

Outline

- Multi threading in MATLAB
- Parallel Pools
- parfor
- spmd
- distributed
- GPU computing
- Cluster Profiles
- MATLAB batch command
- Remote job submission

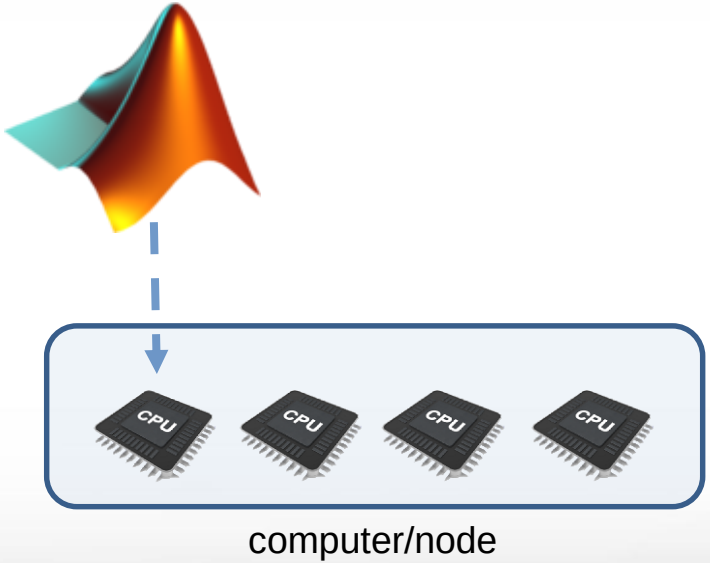
Short course home page:

https://hprc.tamu.edu/training/matlab_parallel_toolbox.html

Matlab source codes:

- On the course home page
- On ada: /scratch/training/MATLAB-PCT/matlab.zip
- On terra: /scratch/training/MATLAB-PCT/matlab.zip

Multi threading

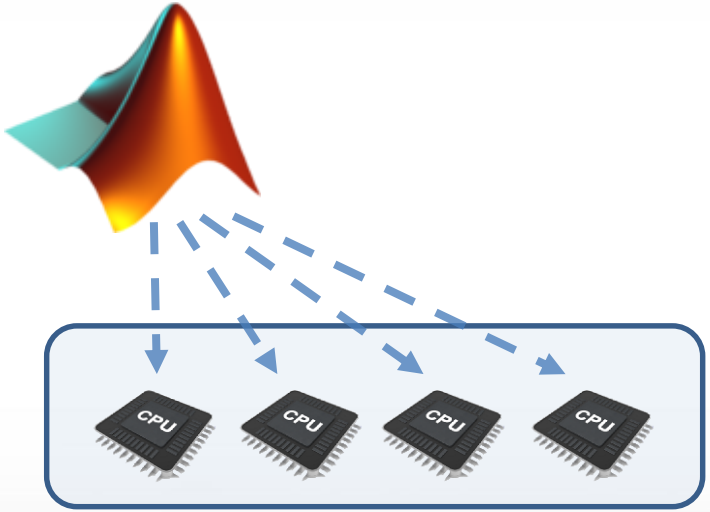


Multi threading

MATLAB automatically executes a large number of operators multi threaded

- Transparent to user
- Array/Matrix operations
- Elementwise operators

```
>> feature('NumThreads',N);
>> old = maxNumCompThreads(N);
```



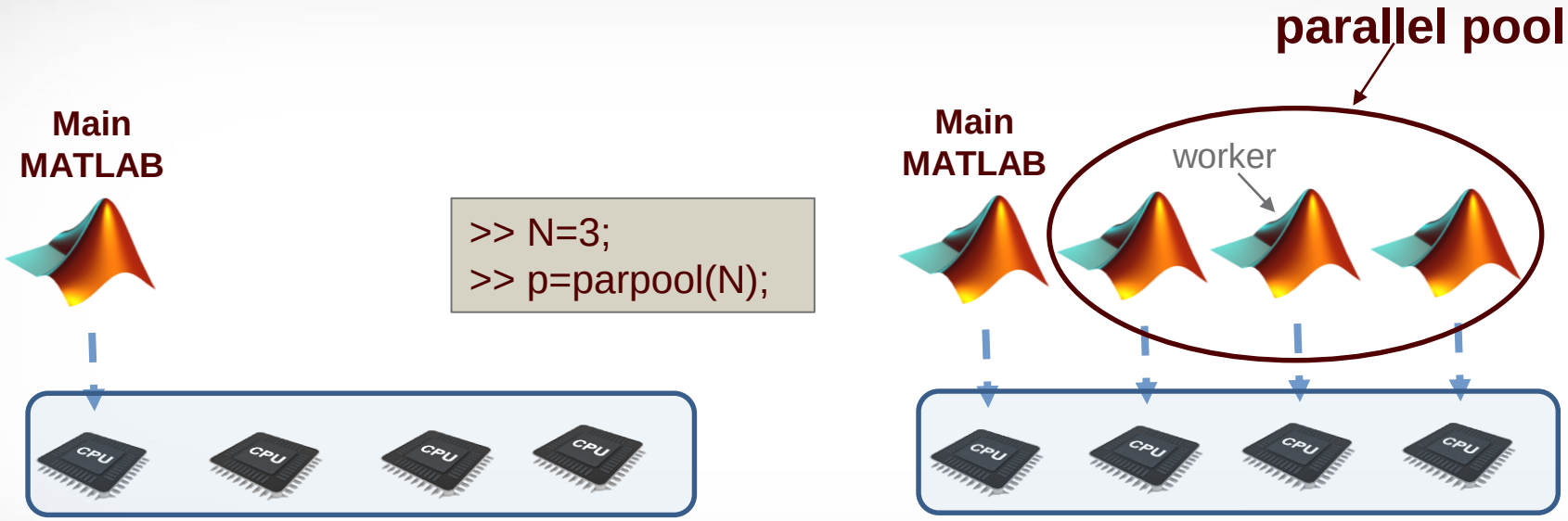
computer/node

HPRC plug

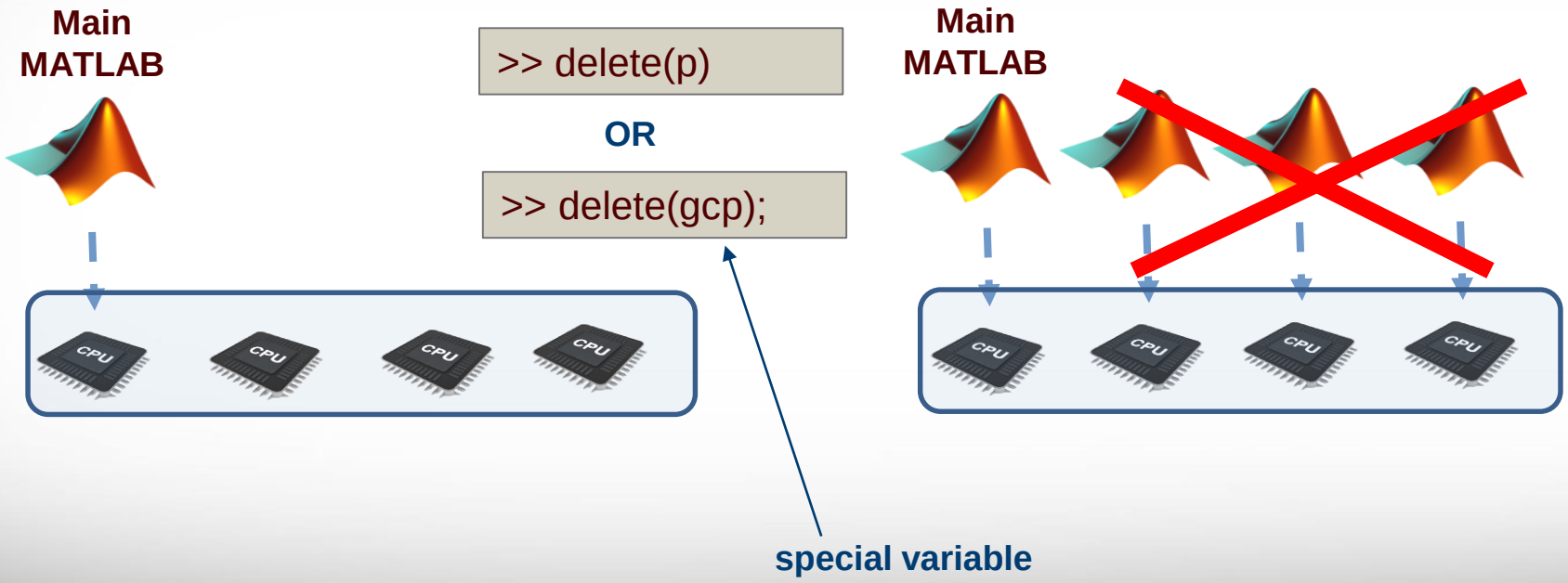
Average Desktop/Laptop has 4 to 8 cores. HPRC cluster terra has 28 cores (20 on ada, some nodes have 40 cores)

Parallel Pool

Parallel Pool



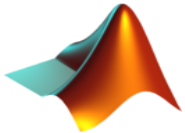
Parallel Pool



parfor

```
for i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```

```
parfor i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```



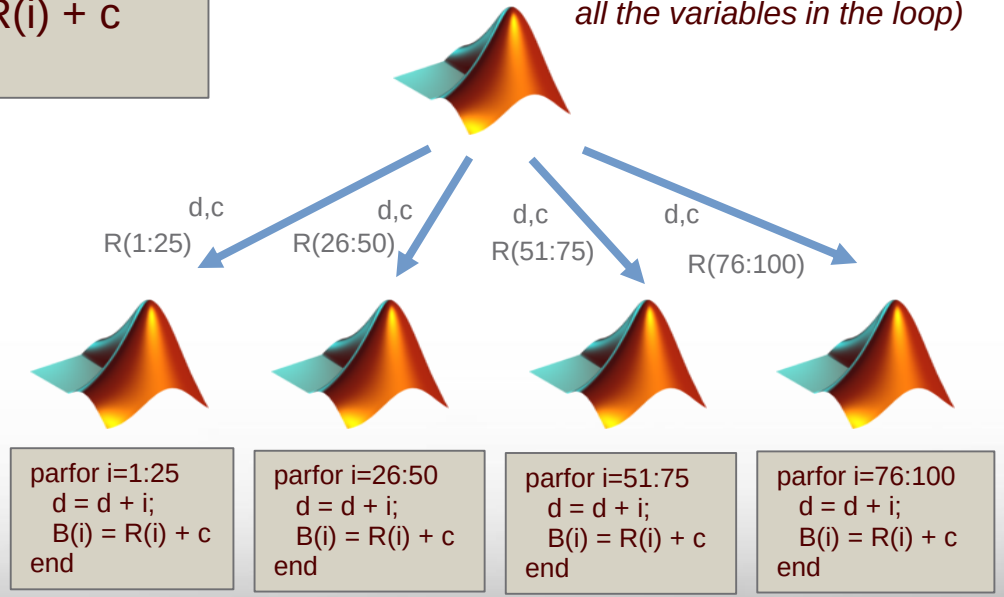
(Workers are idle, waiting for more work to do)

parfor

1) **Main MATLAB sends data and code to workers**

```
parfor i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```

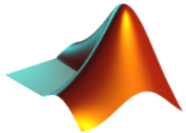
(Before sending data, main MATLAB needs to classify all the variables in the loop)



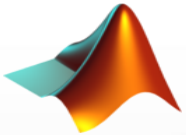
parfor

1) Main MATLAB sends data and code to workers
2) **Workers execute assigned iterations**

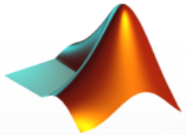
```
parfor i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```



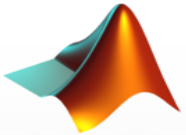
(Main MATLAB waiting for results)



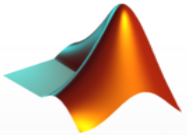
```
parfor i=1:25
   d = d + i;
   B(i) = R(i) + c
end
```



```
parfor i=26:50
   d = d + i;
   B(i) = R(i) + c
end
```



```
parfor i=51:75
   d = d + i;
   B(i) = R(i) + c
end
```

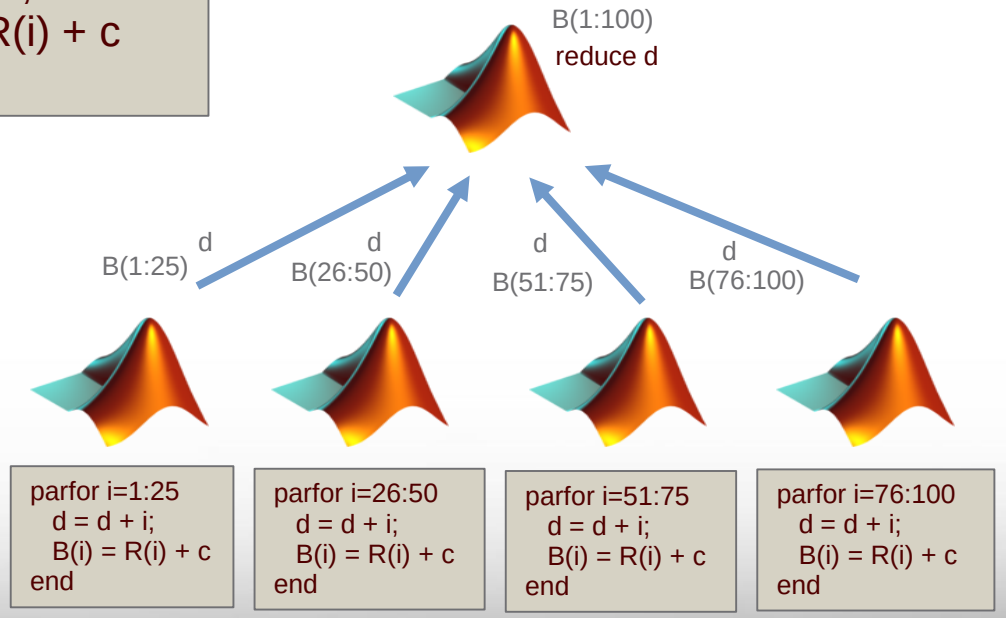


```
parfor i=76:100
   d = d + i;
   B(i) = R(i) + c
end
```

parfor

1) Main MATLAB sends data and code to workers
2) Workers execute assigned iterations
3) **Workers send results back. Main MATLAB combines results**

```
parfor i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```

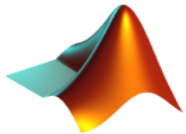


parfor

1) Main MATLAB sends data and code to workers
2) Workers execute assigned iterations
3) Workers send results back. Main MATLAB combines results
4) **Main MATLAB gets control back**

```
parfor i=1:100
   d = d + i;
   B(i) = R(i) + c
end
```

(continues executing statements after parfor)



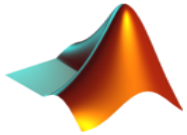
(Workers will be idle again, waiting for more work to do)

SPMD

SPMD block is a construct where all the workers will execute the code in the SPMD block concurrently.

```
spmd
   id = labindex;
   tot = numlabs;
   a=ones(tot)*id;
end
a1 = a{1};
```

special variables

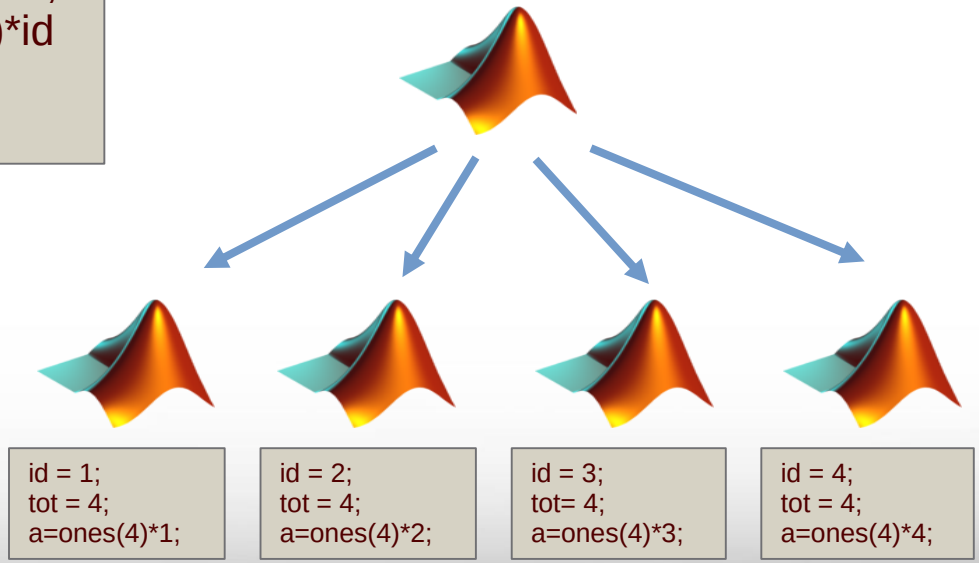


(Workers are idle, waiting for more work to do)

SPMD

1) **Main MATLAB sends code block to workers**

```
spmd
   id = labindex;
   tot = numlabs;
   a=ones(tot)*id
end
a1 = a{1}
```

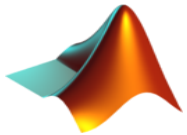


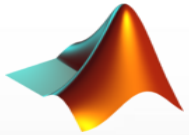
SPMD

1) Main MATLAB sends code block to workers
2) **Workers execute code in SPMD block**

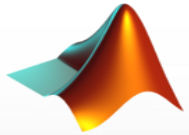
```
spmd
   id = labindex;
   tot = numlabs;
   a=ones(tot)*id
end
a1 = a{1}
```

(Main MATLAB waiting

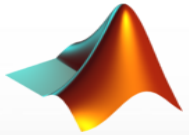




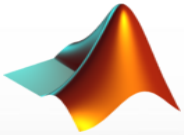
```
id = 1;
tot = 4;
a=ones(4)*1;
```



```
id = 2;
tot = 4;
a=ones(4)*2;
```



```
id = 3;
tot= 4;
a=ones(4)*3;
```

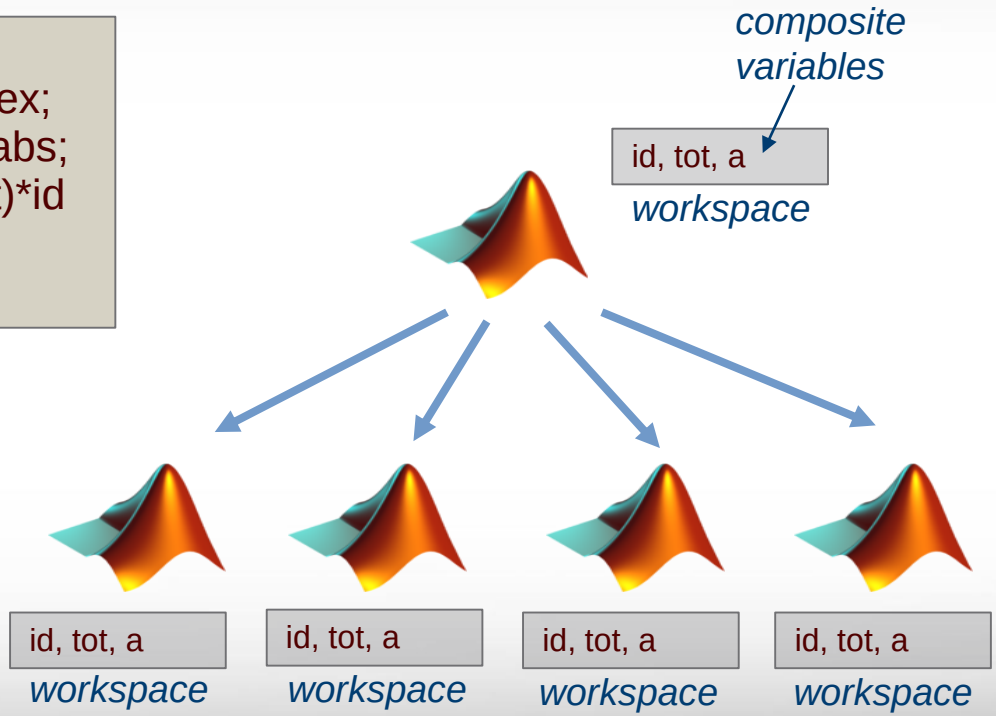


```
id = 4;
tot = 4;
a=ones(4)*4;
```

SPMD

1) Main MATLAB sends code block to workers
2) Workers execute code in SPMD block
3) **Main MATLAB gets control back**

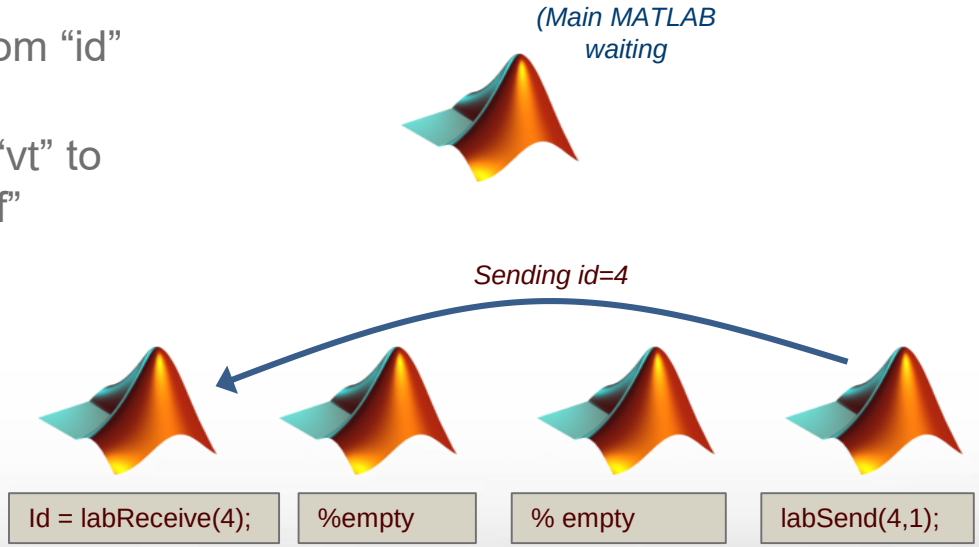
```
spmd
   id = labindex;
   tot = numlabs;
   a=ones(tot)*id
end
a1 = a{1}
```



SPMD Communication

- labSend(var,id) → sends variable “var” to worker “id”
- var=labReceive(id) → receives data from “id” and assigns to “var”
- vf=labSendReceive(it,if,vt) → sends “vt” to “it”,receive data from “if” and assign to “vf”

```
spmd
   id=labindex; n=numlabs;
   if  (id == n)
      labSend(1,id);
else if (id == 1)
    id=labReceive(n);
end
```

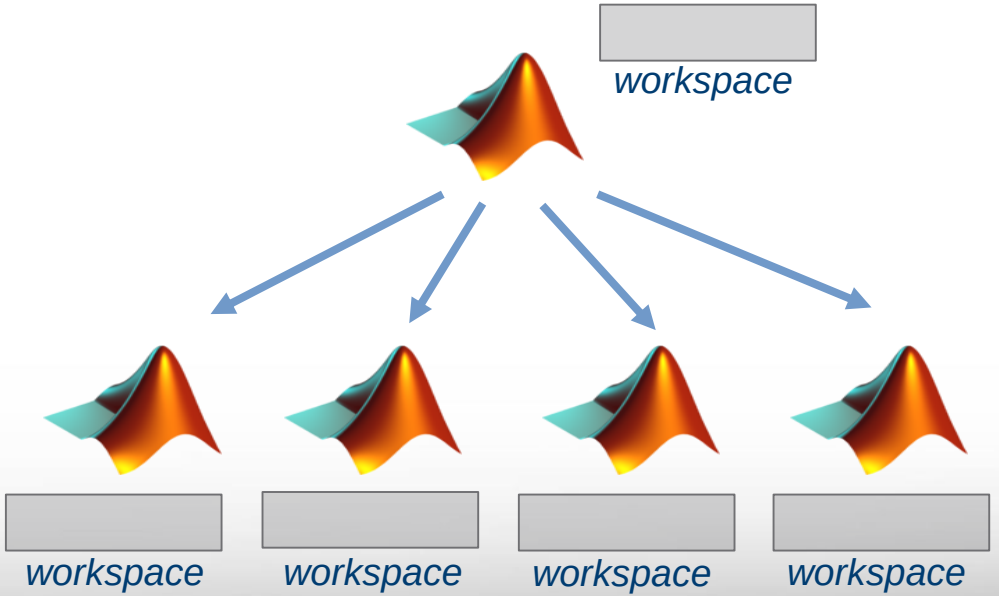


Distributed Arrays

Conceptually very similar to a regular array.

- Many regular matrix operators available for distributed arrays.
- Elements can be of any type
- Elements distributed over the workers
- **Matlab will automatically use parallel version of operator If operand is distributed variable**

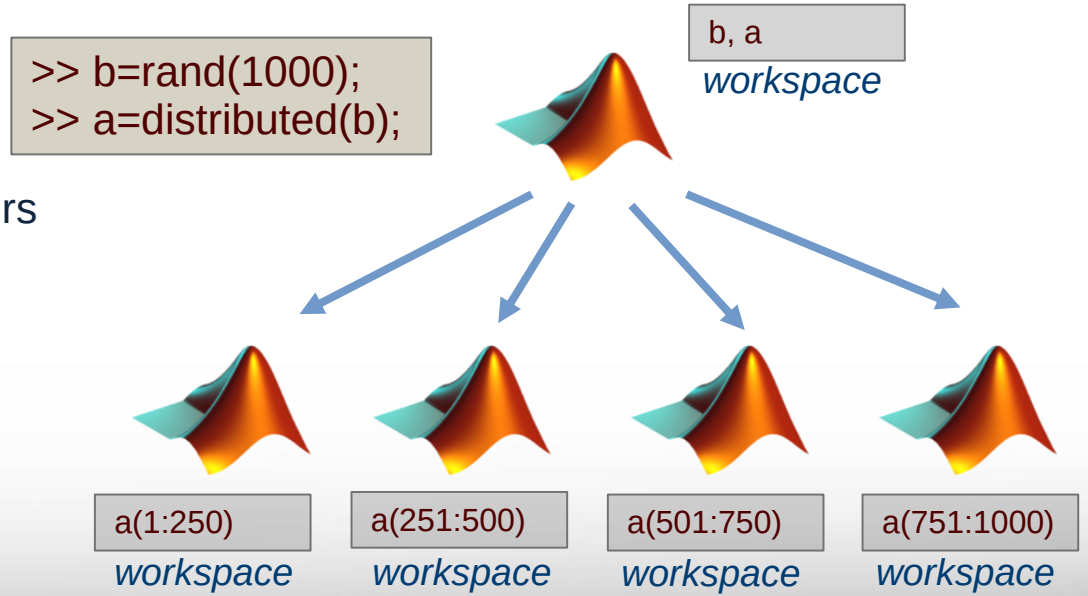
```
>> methods('distributed');
```



Distributed Arrays

Conceptually very similar to a regular array.

- Many regular matrix operators available for distributed arrays.
- Elements can be of any type
- Elements distributed over the workers
- **Matlab will automatically use parallel version of operator If operand is distributed variable**



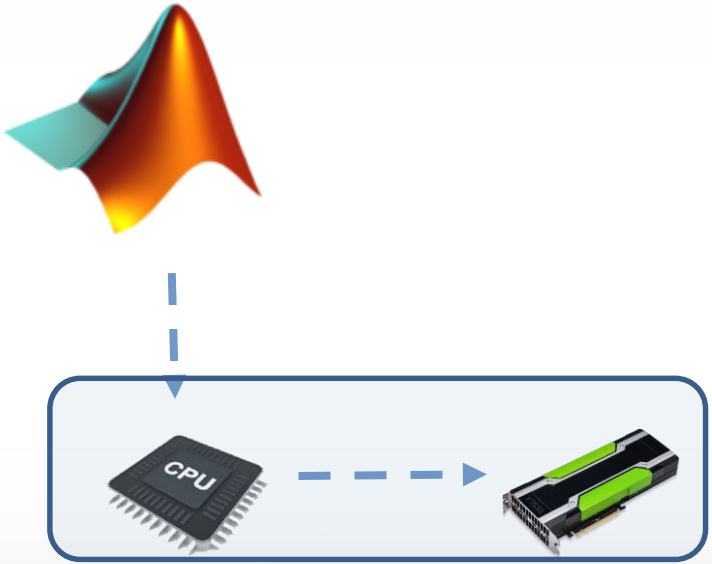
GPU Programming

What is a GPU?

- Accelerator card
- Thousands of small computing cores
- Dedicated high speed memory.

GPU Programming

- MATLAB provides GPU versions for a large number of MATLAB operators.
- Completely transparent to user
- **Matlab will automatically use GPU version of operator If operand is GPU variable**

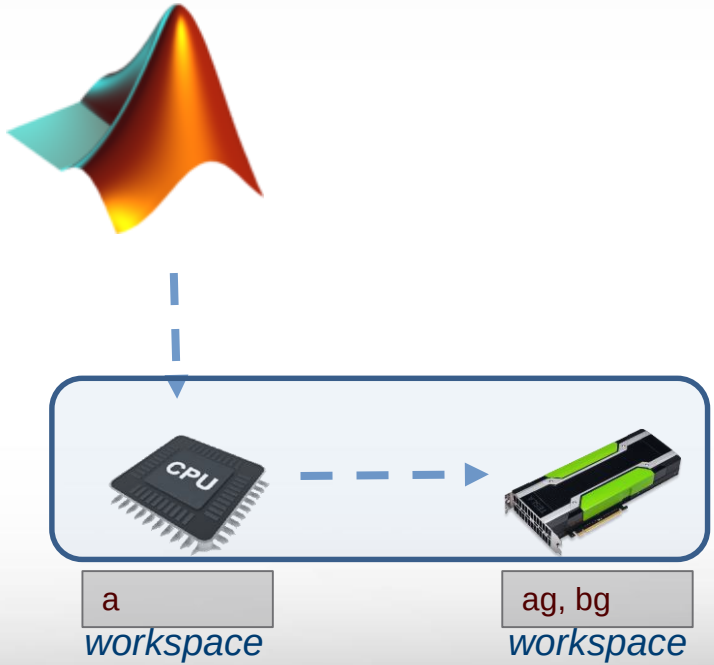



```
>> methods('gpuArray');
```

Copy to/from GPU

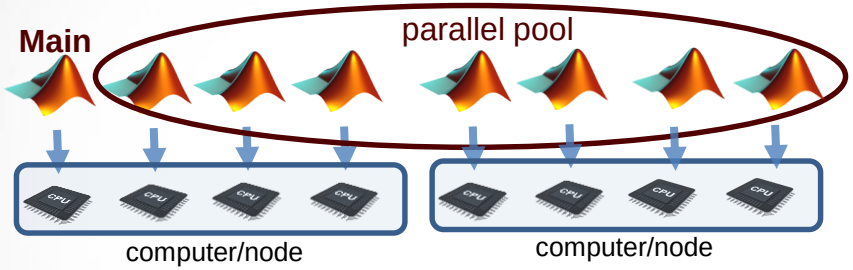
- function: var2=gpuArray(var1) → copy variable “v1” on host to GPU and name “v2”
- function: var2=gather(var1) → copy variable “v1” on GPU to host and assign to variable “var2”
- Use convenient functions to create data directly on the GPU

```
>> a  =  zeros(100);
>> ag = gpuArray(a);
>> bg = gpuArray.rand(100);
```



parpools revisited

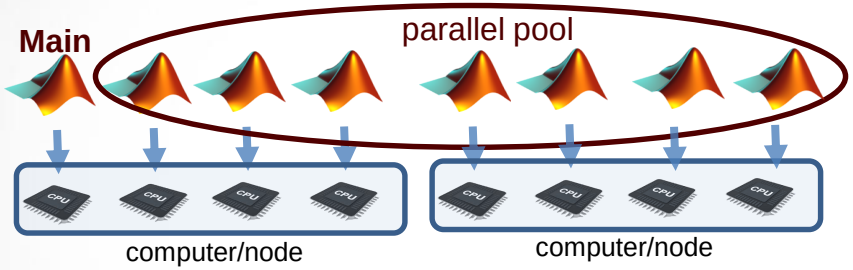
What if you want more workers than cores/nodes?



Remember from creating parpool, didn't provide cluster profile, so using default '**local**' profile!

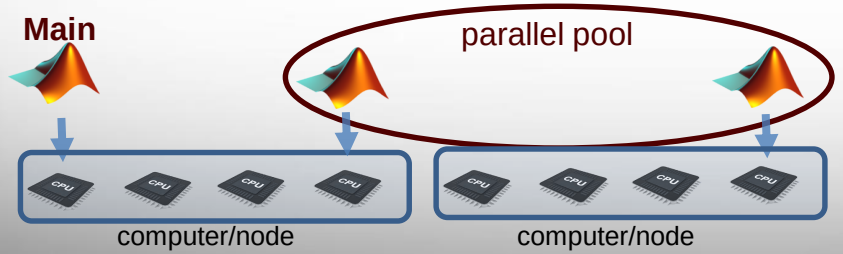
parpools revisited

What if you want more workers than cores/nodes?



or want to distribute workers over multiple nodes?

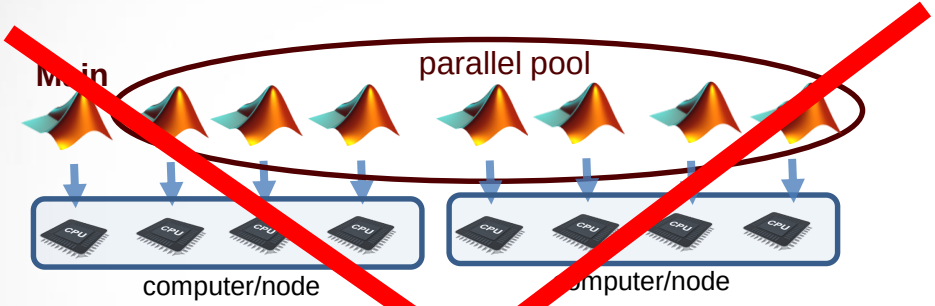
(e.g. so each worker can use more threads)



Remember from creating parpool, didn't provide cluster profile, so using default '**local**' profile!

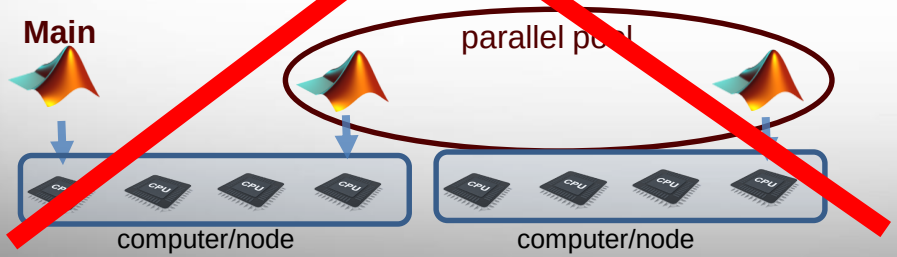
parpools revisited

What if you want more workers than cores/nodes?



or want to distribute workers over multiple nodes?

(e.g. so each worker can use more threads)



Local Profile:

- Workers must run on same computer/node as main MATLAB
- Limited to the number of cores on a computer/node → 28 workers on terra (20 on ada)
- Part of the MATLAB Parallel Toolbox

Only on HPRC

Cluster Profile

- Number of workers only limited by license (currently 96, shared)
- Integrates with Batch scheduler (e.g. slurm and lsf)
  - Will actually submit lsf/slurm jobs
  - Workers will be running on the compute nodes

MDCS license

Only on HPRC

Importing Cluster Profile

```
>> profile = parallel. importProfile(<PATH>);
```

- Only need to import cluster profile once
- Pre-created profile located in $MATLABDIR/profiles/TAMU

tamu convenience function:

```
>> tamu_import_TAMU_clusterprofile();
```

- Wrapper around parallel.clusterProfile()
- No need to provide location of pre-created profile
- Creates directory in scratch directory to store meta data

Only on HPRC

Cluster Properties

How to attach properties (e.g. workers/threads/time)?

Defining cluster properties:

```
>> help TAMUClusterProperties % to see all options
>> tp = TAMUClusterProperties();
>> tp.workers(4);
>> tp.walltime('02:00);
```

HPRC Developed

Attaching properties to cluster profile:

```
% attach the properties to the profile
>> profile= tp.tamu_set_profile_properties(tp);
```

MATLAB batch function

Offloads a script or function to worker(s), control is returned immediately, Job object is returned

```
>> j=batch(cluster obj>,'myscript','Pool',N);  % offloads script (start pool)
>> j=batch(<cluster obj>, @myfunc,N,{x1,x2}); % offloads function
```

```
>> j= tamu_run_batch(tp,'myscript');
```

HPRC Developed

Retrieving Job info and results:

```
>> r= j.State(); %
>> j.wait();  % offloads script (start pool)
>> j.load(); % offloads script (start pool)
>> res= j.fetchOutputs(); % offloads function
```

HPRC Developed

Remote batch submission

Submit jobs from user's local MATLAB session (laptop/desktop)

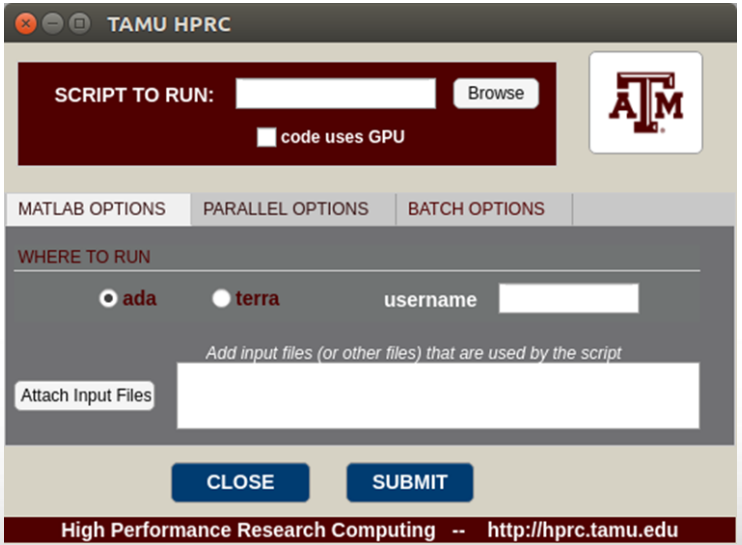
from Matlab command line

```
>> tp = TAMUClusterProperties();
>> tp.hostname('terra.tamu.edu');
>> tp.user('<netid>');
>> j1=tamu_run_batch(tp,'mytest');

% or run a function

>> cp=tamu_set_profile_properties(tp);
>> j2=batch(cp,@myfun,1,{a,b});
```

using the MATLAB app



Download framework and app at https://hprc.tamu.edu/files/HPRC.mlappinstall

https://hprc.tamu.edu/wiki/SW:Matlab#Running_.28parallel.29_Matlab_Scripts_on_HPRC_compute_nodes

Questions?

For additional information/help:
See Wiki: https://hprc.tamu.edu/wiki/SW:Matlab
Send email: help@hprc.tamu.edu