Matlab on multi-processor machines

The aim of this short guide is to get you started with using matlab in a multi-processor environment. The goal is to instruct matlab to use a `pool' of n processors rather than a single one to speed up calculations.

 Before trying to parallelize any codes, please make sure your code works as expected on one processor.

 matlab versions prior to 2007 do not have the features discussed here.

Parallel Computing Toolbox aka matlabpool

matlabpool can be used to parallelize your m script across multiple cores in a single machine.

Open matlab and initialize the parallel environment

```
matlab -nodesktop

>>  matlabpool ( 'open', 'local', 8)
Starting matlabpool using the 'local' configuration ... connected to 8 labs.
>> p=findResource('scheduler', 'configuration', 'local');

p =

Local Scheduler Information
===========================

                      Type : local
              ClusterOsType : unix
                ClusterSize : 8
               DataLocation :
/home/lenocil/.matlab/local_scheduler_data/R2010b
       HasSharedFilesystem : true

- Assigned Jobs

           Number Pending  : 0
           Number Queued   : 0
           Number Running  : 1
           Number Finished : 0

- Local Specific Properties

         ClusterMatlabRoot : /software/matlab/matlab-R2010b
```

The matlabpool cmd instructs matlab that you intend to run your program in on multiple processors. In the example above you are using a parallel environment called `local' (this is usually the default. See below on how to change its properties or create a custom one.) requesting 8 processors. `local' allocates resources that can be listed using the cmd `findResource`.

At this point you can start programming in matlab as usual. To make sure you can fully utilize the resources that matlabpool allocates, do not forget to write your loops using `parfor`. Once finished remember to close matlabpool

```
matlabpool('open',4); % opens the default parallel environment, usually
called `local'
parfor k = 1 : 100000
    some code
end
matlabpool close % release the resources
```

Managing parallel configurations

The `local' configuration is the default. Unfortunately this configuration imposes limits on the numbers of workers available to the matlabpool aka Parallel Computing Toolbox and no Matlab Distributed Computing Server (MDCS) installed. In matlab2010 the max number of workers is set to 8 where as that limit ip brought to 12 in matlab R2011b.

If you wanted to use 64 workers (for instance on one of the maris nodes), you will need a 64-node MDCS licence (you can buy it through our procurement office) and you will also need to create a `scheduler' to manage these resources. There are detailed instructions on how to do this at http://nl.mathworks.com/support/product/DM/installation/ver_current/.

To modify the local configuration

```
jm=findResource('scheduler', 'configuration', 'local');
jm.ClusterSize=6
jm.DataLocation="/somewhere/you/like"
```

To check what configurations are available and which one is loaded use

```
[conf, allConf] = defaultParallelConfig
conf =

test


allConf =

    'local'    'test'
```

Gotchas

- `parfor' executes loop in an order that suits matlab, that is not sequentially
- `parfor' loops cannot be nested

Useful readings

http://vtchl.uiuc.edu/sites/default/files/MATLAB_Report.pdf

http://www.mathworks.com/support/product/DM/installation/ver_current/Files/mdcs-mjs-quickstart-guide.pdf

http://nl.mathworks.com/help/distcomp/program-independent-jobs-for-a-generic-scheduler.html