Experiences with Distributed and Parallel MATLAB on a Microsoft CCS cluster

Daniel Goodman, Stef Salvini & Anne Trefethen

The CCS Cluster

- 20 dual CPU, dual core SMP nodes with 8 GB of RAM
- 2 dual CPU, quad core SMP nodes with 32 GB of RAM
- Gigabit private network
- 10 terabyte file store
- Installed libraries include;
- MS-MPI, Intel Math Kernel Libraries, Numerical Algorithms Group Windows libraries and ITK
- 32 Distributed MATLAB® licenses

Users

- Medical Imaging Applications
- Analysis of Electron Microscope Data
- Modelling residual stress in materials

Distributed MATLAB

- Allows multiple instances of MATLAB to run as workers on clusters.
- These workers can be used to run a range of different styles of job. (trivially parallel, message passing, global operations)
- Supports a set of distributed matrices that can be used to abstract the parallelisation from the system.

Independent Tasks

Below is the code we implemented to run a set of independent tasks on the cluster. The existing code is first re-factored by hand into a single function 'projective_reconstruction_core'. Then standard distributed toolbox objects and tools developed by OeRC are used to create multiple tasks that execute the function on the cluster. As the tasks are independent and potentially numerous, the scheduling can be very efficient, however results have to be returned to the client for marshalling and saving, potentially making a bottle neck, especially if the client is on a slow network connection.

```
jm = findResource('scheduler','configuration', 'CCS')

job = createJob(jm);

for z_block_start=1:30
    createTask(job, @projective_reconstruction_core, 1,
    {imodfile_in, numtlts, xsize, ysize, plane_coeffs2,
    blocksize, z_inc, homography_3D, z_block_start});
end

f = calculateFiles('projective_reconstruction_core.m');
set(job, 'FileDependencies', f)

submit(job)

waitForState(job, 'finished')

blocks = getAllOutputArguments(job);
```

Positive	Negative
• Potential to efficiently schedule heterogeneous tasks • Easy to create heterogeneous tasks	• All results returned to clients machine • Fiddly control of returned results

Experiences with CCS

- The MS CCS cluster is easy to use out of the box solution for mid-size computing needs.
- It provides a parallel computing system in a windows environment which is becoming increasingly important for many research areas
- For the most part the submission of jobs, the job scheduler, and the system in general is very accessible for the non-expert user.
- Managing licenses is tricky. License conditions are currently checked by Submission and Activation filters. These filters are implemented by single executables, making management on a cluster hosting heterogeneous applications hard. There is also no mechanisms to allow the activation filter to feed back to the scheduler why the job was rejected.
- It can also be hard to determine which application is attempting to run, as the only information is the command to be executed, and this executable can belong to CCS taking the application executable as an argument.
- We had difficulties because of the need to copy large data files to nodes, where on occasion due to a time out the file server ceases to appear as a network resource, resulting in transfers failing.
- Debug tools were not available in the version of CCS we used. Often when a job failed, no error message was provided to assist in debugging. Console output had to be retrieved by hand from log files.
- Also on some client machines it is not possible to get the client to remember the user's password and automatically authenticate. This makes running scripts that pass some tasks to the cluster difficult without the user being present.

Communicating Tasks

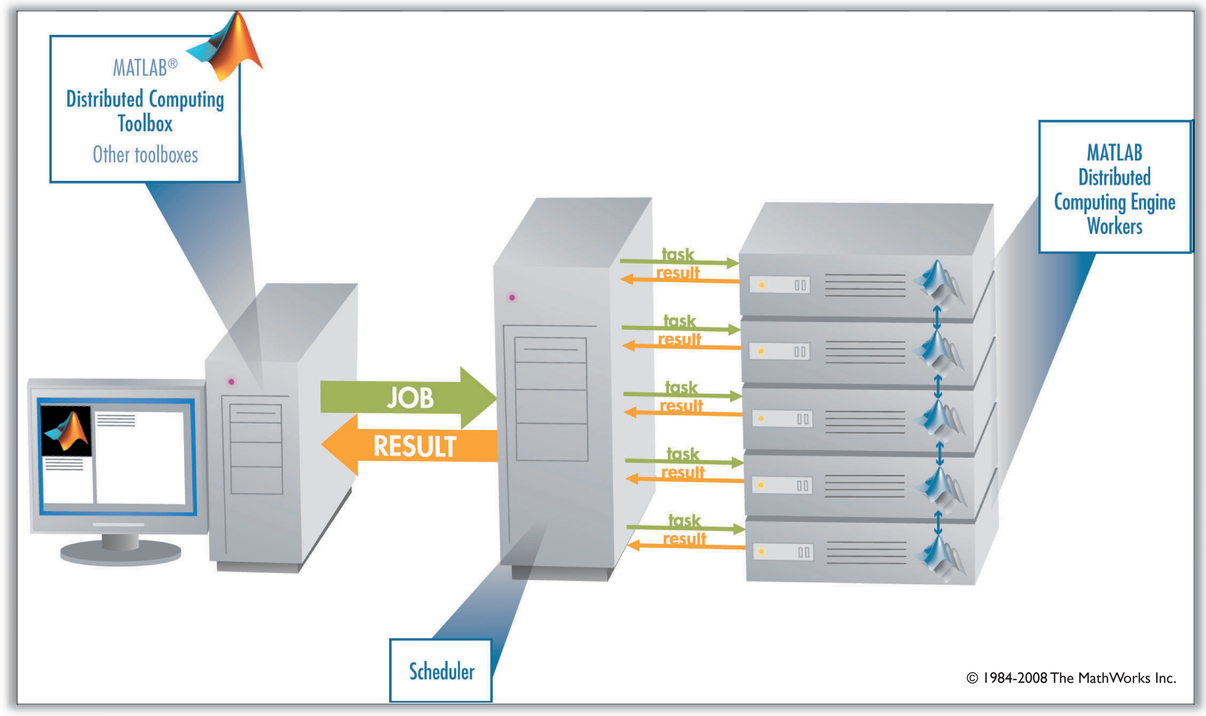
The main alternative to independent tasks is the communicating tasks implemented below Here there is a single task that has multiple instances which run concurrently and can communicate. Doing this they are able to pass a token round controlling who is next to write to disk. This allows them to save directly to the file store. As with the independent tasks, the code is first re-factored to a single function that the task will execute. However, the code is also augmented with calls to `LabSend` and `LabReceive` in order to pass the token, and code to determine which piece each instance should address, based on its `labindex`. OeRC have produced a set of constructs to encapsulate this to ease the programming of such functions. Once this was done standard distributed toolbox objects and tools created at OeRC are once again used to construct the task and execute it on a set of cluster nodes.

```
sched = findResource('scheduler', 'configuration', 'CCS');

pjob = createParallelJob(sched);

set(pjob, 'MaximumNumberOfWorkers', 30)
set(pjob, 'MinimumNumberOfWorkers', 15)

f = calculateFiles('projective_reconstruction');
set(pjob, 'FileDependencies', f)

task = createTask(pjob, @projective_reconstruction, 0,
{basename});

submit(pjob)

waitForState(pjob)
```

Positive	Negative
• Data can be easily written back to the File Store	• Lack of multi-threading for tasks with heterogeneous execution times wastes resources.

Experiences with MATLAB

- MATLAB and the distributed toolbox are easy to install, configure and indeed use.
- The distributed toolbox brings a higher level interface to parallel computing and makes parallel computing accessible to a broader group of users.
- Creating distributed and parallel applications can now be achieved in minutes.
- To provide a simpler interface we have developed a set of tools so that users don't need to explicitly declare configuration parameters such as which .m files need to be transferred or which path is to be used to gain access to the function code.
- It would be easier if there were tools to re-factor code into the desired single function to be called by a task, at present this is done by hand.
- The integration of tools to enable automated setting of configuration parameters would make using the system less messy and again easier to use.
- In our application it would have been useful to have the capability load a partial piece of a data file. However, currently much of the data model is built around the use of .MAT files to transmit data which leads to the whole of sometimes very large files being sent to and loaded by each instance of MATLAB (possibly many per node).
- The ability to have multiple threads sharing data or instances of MATLAB would have made our application more efficient. At present the model has as many MATLAB processes as there are cores, all sharing the same cache, but all requiring that any data is separately transmitted to them over the network.



The MathWorks Microsoft® oerc UNIVERSITY OF OXFORD