Ecce Compute Server Registration

Overview

Ecce supports job submission to a wide range of machines from workstations to supercomputers or clusters running queuing management software including Maui, LoadLeveler, NQS, NQE, and PBS. With a little work and testing, support for other systems can be added by any Ecce installation. This should be a fairly simple task for systems similar to the ones named above such as LSF and EASYMCS. However, since we do not have access to resources running this software, we do not support them directly but instead provide instructions on how to add support. Ecce also supports job submission to machines running Globus 1.2 services.

In order for Ecce to automate job submission and monitoring, information about compute resources and installed computational software must be registered with Ecce. This is currently true even for machines running the Globus software. Registering this information requires one or two steps:

1. Provide information to the Machine Registration GUI tool (GUI throughout this document). The GUI is shown in the next section and described in the sections GUI Users's Guide and GUI Field Information.

2. Customize interaction with the queueing system by editing some text files. This step is typically not needed for workstations. The process consists of customizing queue header template information as well has program invocation (poe, prun...). There are additional loop-holes in place for customizing the execution environment if needed.

Once a machine is registered to Ecce, the information is used to:

- perform validation checks against user input prior to job launching
- generate job submission scripts
- connect to and submit jobs
- monitor the state and output data from jobs
- track runtime data pertaining to job runtime performance. The latter can be used to predict job costs once a sufficient amount of data has been collected over time.

Registration can take place in two ways:

- Individual workstations can be registered by each user through the Job Launcher tool. In fact, when starting the Calculation Manager for the first time, the local machine is automatically registered with information that can be extracted from the operating system. However, the user must finish the registration process to provide the installation location of any chemistry codes they plan to run. This can be completed entirely through the Launcher machine registration GUI.

- Shared resources which typically run queue systems are registered by an Ecce administrator. Registering a queued machine is more complex than registering a workstation since information on queues must be provided along with customized headers for the queue system and customized commands to start the job execution (i.e., poe or mpirun...)

This document is primarily intended for Ecce administrators registering widely shared resources. It provides a brief user's guide to get you started, describes each of the GUI fields, provides hints for finding some of the information for your queue system, and finally, discusses how to do customization not supported through the GUI, but typically required for queued systems.

GUI User's Guide

The GUI allows an Ecce administrator to register a new machine, modify the registration information for a machine, or delete a registered machine. This guide is intended to help the Ecce Administator manage registration using the GUI. However, as mentioned in the overview, some editing of files will also be required.

Starting the GUI

To start the machine configuration gui as an ecce administrator, type the following at the command line prompt:
ecce -admin

The GUI below will be displayed.

Please Note: The GUI does *NOT* provide input validation by checking your input against the actual machine configuration. Please be sure your registration information is accurate, otherwise the operation of Ecce will be impaired. For example, it will let you specify a minimum number of nodes that is greater than the maximum number of nodes.

Adding a New Machine

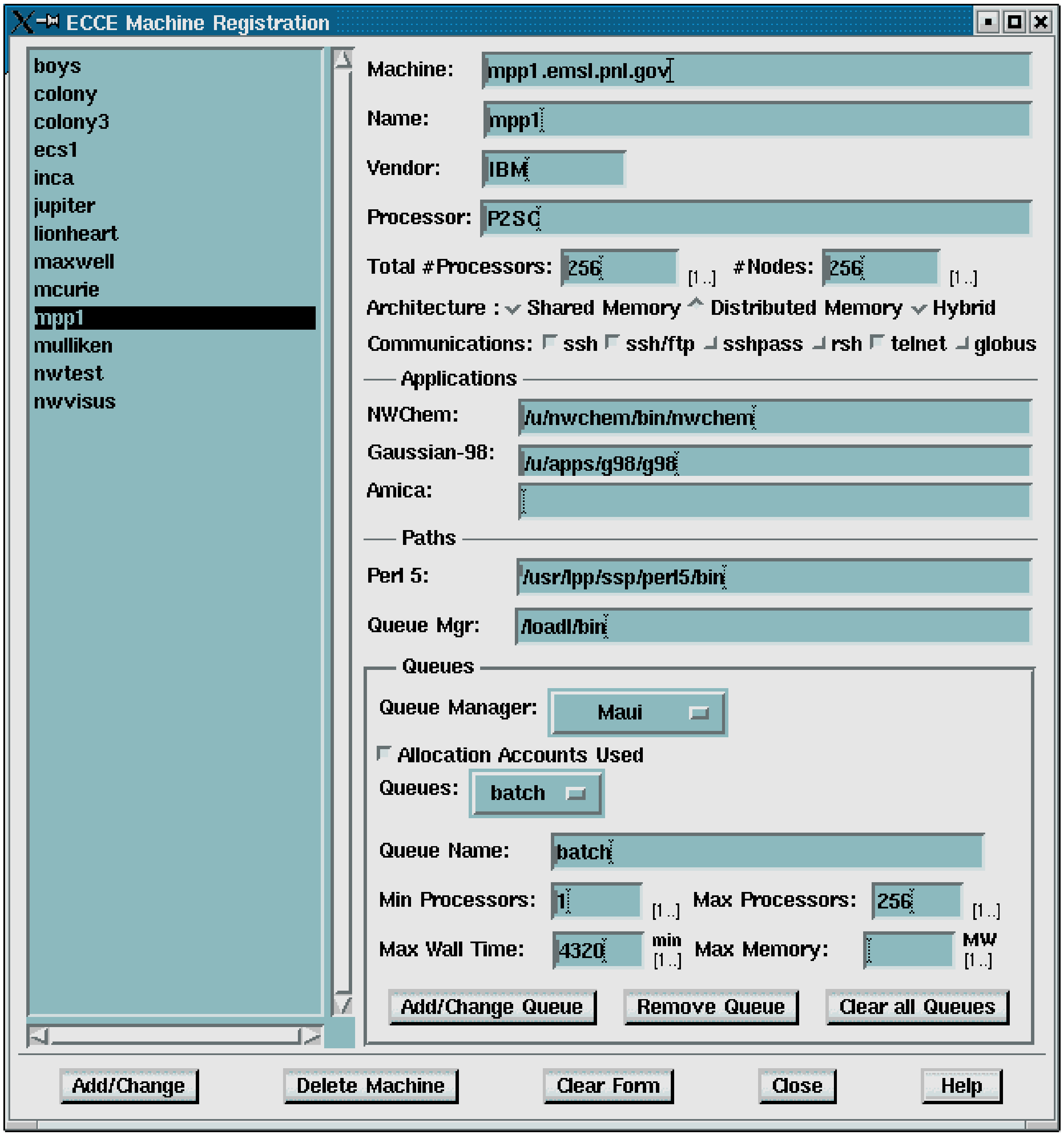
To register a machine with Ecce from scratch:

1. Hit the **Clear Form** button to reset the entire form.

2. Type in the name of the new machine in the **Name** field. The name must include the full domain name. For example, mpp1.emsl.pnl.gov.

3. Type in registration information. Descriptions of the meaning of each field is provided in the section GUI Field Information.

4. Hit the **Add/Change** button to save your changes.


ECCE Machine Registration

boys
colony
colony3
ecs1
inca
jupiter
lionheart
maxwell
mcurie
mpp1
mulliken
nwtest
nwvisus

Machine: mpp1.emsl.pnl.gov

Name: mpp1

Vendor: IBM

Processor: P2SC

Total #Processors: 256 [1..] #Nodes: 256 [1..]

Architecture : Shared Memory [✓ Distributed Memory] Hybrid

Communications: ssh ssh/ftp sshpass rsh telnet globus

Applications

NWChem: /u/nwchem/bin/nwchem

Gaussian-98: /u/apps/g98/g98

Amica:

Paths

Perl 5: /usr/lpp/ssp/perl5/bin

Queue Mgr: /loadl/bin

Queues

Queue Manager: Maui

Allocation Accounts Used

Queues: batch

Queue Name: batch

Min Processors: 1 [1..] Max Processors: 256 [1..]

Max Wall Time: 4320 min [1..] Max Memory: MW [1..]

To register a machine that has a similar configuration as an existing registered machine:

1. Highlight the name of the existing machine in the machine list box.

2. Type in the name of the new machine in the **Name** field. The name must include the full domain name. For example, mpp1.emsl.pnl.gov.

3. Type in registration information making any changes as necessary. Descriptions of the meaning of each field is provided in the section GUI Field Information.

4. Hit the **Add/Change** button to save your changes.

Please Note: Only the buttons at the very bottom of the form cause changes to be saved. The buttons within the **Queue panel** are only intended change certain field values in the GUI and do not save data. When you are finished editing Queue information you must use the buttons at the bottom of the form to save machine registration information.

Modifying a Machine Configuration

1. Select the machine you intend to edit from the list on the left.

2. Type in registration information making any changes as necessary. Descriptions of the meaning of each field is provided in the section GUI Field Information.

3. Hit the **Add/Change** button to save your changes.

Deleting a Machine

1. Select the machine you intend to edit from the list on the left.
2. Hit the **Delete Machine** button to delete the registered machine.

GUI Field Information

This section describes each data entry field representing machine registration information and provides a list of mandatory fields required for each type of machine. By convention, each data entry field in the GUI appears in bold letters.

Name

This is the machine name, and it must include the full domain address. For example mpp1.emsl.pnl.gov.

Vendor, Model, Processor

These fields represent the vendor, model, and processor of the machine. As mentioned in the overview, this information is kept to support future queuries and to enable advisor-type capabilities that can either predict the cost of a calculation and/or choose an appropriate resource based on user constraints. Note that since this data is for future queuries, you may leave them blank without affecting operations.

Table 1. Describes Fields That are Mandatory, Optional, or NA Depending on the Type of Machine That is Being Registered

GUI Field Name	**Workstation**	**Queued Systems**
Allocation Accounts Used	NA	Mandatory.
Applications NWChem/Gaussian98™	Mandatory for each code that will be used.	Mandatory for each code that will be used.
Architecture	Mandatory for Gaussian98™ script generation.	Mandatory for Gaussian98™ script generation.
Communications	Mandatory	Mandatory.
Max Memory	NA	Optional depending on how queues are managed.
Max Wall Time	NA	Optional depending on how queues are managed.
Min Nodes	NA	Optional depending on how queues are managed.
Model	Optional	Optional.
[Machine] Name	Mandatory	Mandatory.
Paths Perl 5	Mandatory	Mandatory.
Paths Queue Mgr	NA	Mandatory.
Processor	Optional	Optional.
Queue Manager	NA	Mandatory.
Queue Name	NA	Mandatory.
Total #Processors	Mandatory	Mandatory.
Vendor	Optional	Optional.
# Nodes	always 1	Mandatory.

The source of the model and processor information varies depending on the platform. Some helpful hints for selected platforms:

Vendor	**Info**	**Command**	**Example Output**	**Notes**
Sun	model	`uname -I`	SUNW,Ultra-2	Name of hardware implementation.
IBM	model	`uname -m`	000007411C00	Then use man page to decode model ID 1C to model 7013/550.
SGI	proc	`hinv`		MHZ line gives board speed, CPU Line gives chip.

If you wish to provide accurate values, contact your system administrator for help.

Total #Processors

Modern systems often contain many nodes and also many processors per node. The value entered here should be the total number of processors available on the system. For example, if you have a system with 8 nodes and 4 processors per node, you would enter 32 (8*4). If you have a system with 4 nodes, 4 processors per node and 4 additional nodes with 2 processors per node, you would enter 24 (4*4 + 4*2). For workstations, enter the number of CPUs.

# Nodes

This is the total number of nodes on the machine. For example, if you have a system with 8 nodes and 4 processors per node, you would enter 8. If you have a system with 4 nodes with 4 processors each and 4 additional nodes with 2 processors each, you would enter 8 (4+ 4). For workstations, this value is always 1.

Architecture

For multi-processor systems, specify the type of memory architecture. Workstations will generally be shared memory architectures. An example of a distributed memory architecture is an IBM SP system with many nodes but one processor per node. A hybrid system would be an IBM SP with multiple processors per node.

Note: This information is used to determine the correct keyword substitutions needed to run Gaussian98™. If it is not correct, the job will not run. If you are not running Gaussian, this field has no affect.

Communications

This field allows you to specify the communications mechanism used for launching jobs and providing interactive connections to the machine. It is possible for a machine to support more than one mechanism in which case the user will decide which to use. Ecce does not supply any of the communications mechanisms. You or your site administrators are responsible for providing this. Luckily, just about every system already has one or more of these mechanisms already installed. Note that whatever protocol will be used must be installed on both Ecce client and compute server machines.

If rsh is used, users are responsible for settting up correct .rhosts files. You will need to consult your system administrators regarding security issues, ssh is the preferred mechanism. Normally the copy command is paired with its shell command (e.g., rsh/rcp ssh/scp...). However, it is possible to specify these independently. This option has been provided for ssh/ftp since scp can be difficult to manage. There is no need to specify this option unless you are having problems with scp.

The Ecce administrator can customize which protocols are available and how they are invoked. This allows the use of kerberos r-commands as well as support for differences between ssh1 and ssh2. See (Gary).

Application Paths

In order for Ecce to start applications on the compute server, it must know the path to the executables. The application panel lists the computational codes that have been integrated into Ecce. Provide the full path to the executable for any applications that will be run at your site. Note that you are responsible for getting and installing each of the computational codes. They are not shipped with Ecce.

Paths

Ecce also requires a couple more pieces of information about each compute server.

Perl 5 The full path to the installation of perl5 is required to support real time monitoring of the jobs and must be provided.

Queue Mgr On systems managed by a scheduler, you must provide the path where the queue system commands are installed. For example for LoadLeveler, this would be the directory where llsubmit and related programs are installed. This information must be provided to successfully submit jobs on queued machines.

Queues Panel

If your system is running a batch scheduler, you need to provide information about the queues and how the queues are managed. These fields are all provided within the **Queues** panel. For workstations or systems without a batch system, make sure these fields are empty by hitting the **Clear** button within the Queues panel.

For system with multiple queues, you can edit and view only one queue at a time. The buttons at the bottom of the Queues panel are used to add, modify, and delete queue information. These buttons act on the single queue currently shown in Queues option menu. Also note that they only affect in-memory data structures. When you are done specifying queue information as well as the other fields, you must use the buttons on the very bottom of the form to make your changes persistent.

Queue Manager

By default, Ecce knows about several queueing systems such as Maui, NQS, PBS, etc and has been configured to support them. If you are using one that is not currently supported, you can add it by following the instructions in the section Adding Queue Manager. It will then show up in this list. Select the correct queue system so Ecce will know which commands to run when submitting and controlling jobs.

Allocation Accounts

Check this box if the queuing system is configured to require the specification of an allocation account name with job submissions. If this box is checked, the user will be required to fill in a field with their account name when launching a job.

Queues

This option menu contains the list of all queues currently configured for the machine. By selecting a queue, the remaining fields in the **Queues** panel are updated to show how the queue is configured. If you make edits to a queue definition and then select another queue without hitting the **Add/Change Queue** button, your changes will be lost. Likewise, if you switch to a different machine without hitting the Add/Change button at the bottom of the GUI your changes will be lost. See the Hints section for some hints on how to find the requested information for some of the queuing systems.

Name

Identify the name assigned to the queue. A typical example is “batch”.

Min Nodes

The minimum number of nodes that must be requested for a job to use this queue. This information is required. A single minimum value may be hard to define due to variable quality of service features that may be in use. A value of 1 is safe in most cases. If there is a minimum smaller than the one you supply in this field, the queue system will generate an error that will be presented to the user when they submit the job.

Max Nodes

The maximum number of nodes that can be allocated to this queue.

Max Wall Time

Enter the maximum amount of elapsed time that a single job can use in this queue. When this limit is reached, the job will be terminated. This information is optional. Leave it blank if it does not apply to your queue. Contact your administrator to determine the established time limit if you don’t know it.

Max Memory

The maximum amount of memory a job in this queue can request. This information is optional; leave it blank if it does not apply to your queue.

Hints

LoadLeveler

To find out what queues are available and how they are configured, use the llclass command. For example:

```
> llclass -l
=============== Class batch ==========
               Name:  batch
           priority:  30
              admin:  accept:root:loadl
          NQS_class:  F
         NQS_submit:
          NQS_query:
     max_processors:  510
            maxjobs:  -1
      class_comment:
   wall_clock_limit:  3+00:05:00, -1
      job_cpu_limit:  3+00:25:00, -1
          cpu_limit:  -1, -1
         data_limit:  -1, -1
         core_limit:  -1, -1
         file_limit:  -1, -1
        stack_limit:  -1, -1
          rss_limit:  -1, -1
               nice:  0
               free:  8
            maximum:  437
=============== Class support ==========
               Name:  support
           priority:  30
      Include_Users:  d39033
                   :  d3e129
                   :  ...
                   :  zzhang
              admin:  accept:root:loadl
          NQS_class:  F
         NQS_submit:
          NQS_query:
     max_processors:  32
            maxjobs:  -1
      class_comment:
   wall_clock_limit:  3+00:05:00, -1
```

```
        job_cpu_limit:  3+00:25:00, -1
            cpu_limit:  -1, -1
           data_limit:  -1, -1
           core_limit:  -1, -1
           file_limit:  -1, -1
          stack_limit:  -1, -1
            rss_limit:  -1, -1
                 nice:  0
                 free:  33
              maximum:  33
```

For the example above, there are two queues name *batch* and *support*. The minimum number of nodes is 1. The maximum number of nodes is the max_processor field. The Ecce Wall Clock limit is the wall_clock_limit field. The total number of nodes is not available.

Typically, the Maui filter is used on top of LoadLeveler. See the following section.

Maui

Maui allows the configuration of per-user privileges and/or restrictions. There is no standard command to find out what these privileges and restrictions are. You will have to consult your system administrator. In particular, ask for minimum number of nodes and what QOS options are supported. These are "Quality Of Service" levels that can be set up per user. Since Ecce does not support registration of per-user information, you should set the limits to the least restrictive settings. Any violations of privileges will be trapped when the job is submitted.

NQS/NQE

NQS/NQE systems typically have many queues configured with various numbers of processors and time limits. NQS/NQE selects a queue based on how many nodes you want to use and how much time you need. In this case, you should register only one "fake" queue perhaps named "limits". Then for all the input fields, specify the maximum allowed for any queue. This will allow the user to select from the full range of possibilities and let NQE do the rest.

You can also configure NQE so that the user can specify the queue. In this case, register each queue just as with Maui/LoadLeveler. Use the qstat -m command to find out which queues exist and what their limits are.

```
> qstat -m
NQS 3.3.0.8 BATCH QUEUE MPP LIMITS
QUEUE NAME    RUN        QUEUE-PE'S   R-PE'S     R-TIME      P-TIME
              LIM/CNT    LIM/CNT      LIMIT      LIMIT       LIMIT
q3p4          8/0        --/0         4          36000       36000
q4p4          6/1        --/4         4          360000      360000
```

```
q1p24                  4/0       --/0          24         600          600
q2p24                  6/0       --/0          24         3600         3600
q3p24                  6/0       --/0          24         36000        36000
q4p24                  4/0       --/0          24         360000       360000
q2p96                  3/0       --/0          96         3600         3600
q3p96                  2/0       --/0          96         36000        36000
q4p96                  1/0       --/0          96         360000       360000
q2p128                 1/0       --/0          128        3600         3600
q1p96                  2/0       --/0          96         600          600
q1p128                 1/0       --/0          128        600          600
q4p128                 1/0       --/0          128        180000       180000
test                   20/0      --/0          4          1500         1500
test8                    15/0    --/0          4          900          900
q1p48                  4/0       --/0          48         600          600
q2p48                  6/0       --/0          48         3600         3600
q3p48                  6/0       --/0          48         36000        36000
q4p48                  4/0       --/0          48         360000       360000
---------------------- --- ---  ------ ------  ------  ------  ------
T3E                    30/1      128/4
---------------------- --- ---  ------ ------  ------  ------  ------
```

The “R-PE’S LIMIT” column represents the maximum number of nodes.
The “R-TIME LIMIT” column represents the CPU Limit.

Queue System Customizations

A high degree of customization is often required to submit jobs to queued machines and Ecce provides some ways for you to do this. However, none of these customizations can be done through the GUI. Rather, you must use a text editor to edit some files. The files are located in the <top>/siteconfig directory where <top> is the installation location of the Ecce client software. These files may contain additional documentation to assist with this process so please read through any comments in the files.

Briefly, the files of interest are:

- CONFIG.<machine>: where machine is the name of the system you are customizing. The name does *not* include the full domain name. This file contains customized queue headers, execution commands, environment information, and potentially arbitrary csh code. This is where almost all custom information will be provided.

- QueueManagers - specifies queueing systems registered with Ecce and commands required for job control for each queue system.

- submit.site - site defaults that are typically overridden per machine by the CONFIG.<machine> file. There are some useful examples in this file that can serve as a starting point. Generally, you should not modify this file but create customizations on a per machine basis by modifying the CONFIG.<machine> files.

Note that CONFIG.<machine> files are created by the GUI. You must register the compute resource through the GUI before performing any customizations.

CONFIG.<machine> Syntax

The syntax of the CONFIG.<machine> files consists of key value pairs with many styles supported including:

- key: value
- key value
- key { value }
- key {
    value...
    ...
    }

The latter format is the only syntax suitable for large chunks of data and is the preferred syntax.

CONFIG.<machine> Variables

During the process of generating a job script, several variables are available for use within the CONFIG.<machine> files. They include:

- $host - server machine name
- $account - allocation account
- $queue - nam of queuee
- $code - name of code
- $totalprocs - total processors
- $nodes - number of nodes
- $ppn - processors per node
- $wallTime - maximum wall time
- $runDir - location where job will be run
- $scratchDir - scratch directory path
- $inFile - input file name
- $outFile - output file name
- $memory - max memory
- $submitFile - name of submit file
- $g98 - path to Gaussian98™

- $nwchem - path to NWChem

The currently bound value for each variable will be substituted when the job script is generated.

Queue Header

The most common customization required is to the script header which contains directives interpreted by the scheduler. There are several examples in the submit.site file. Copy the one most closely matching your system and modify it to reflect how your system is configured and used keeping in mind that the variables specified in the previous section are available. The best way to get the correct header is to get job script examples from people already submitting jobs to the machine.

For example, submit.site may contain the following header for NQS:

```
NQS {
#QSUB -mb
#QSUB -me
#QSUB -s /bin/csh
#QSUB -nr
#QSUB -lM 28Mw
#QSUB -lm 28Mw
#QSUB -J m
#QSUB -lT 7200  # per request limit
#QSUB -lt 3600  # per process limit
#QSUB -q $queue
#QSUB -eo -o nqe.out
#QSUB -l p_mpp_t=$wallTime
#QSUB -l mpp_t=$wallTime
#QSUB -l mpp_p=$nodes
#QSUB -r $submitFile
#QSUB -A $account
#QSUB
}
```

If your machine doesn't use allocation accounts and specifies memory limits, you can enter the following into CONFIG.<machine> to override the default:

```
NQS {
#QSUB -mb
#QSUB -me
#QSUB -nr -nc
#QSUB -s /bin/csh
#QSUB -nr
#QSUB -lM $memoryMw
#QSUB -lm $memoryMw
#QSUB -J m
#QSUB -q $queue
#QSUB -eo -o nqe.out
#QSUB -l mpp_p=$nodes
#QSUB -r $submitFile
#QSUB
}
```

Code Execution Commands

As mentioned above, you may also need to customize the command that actually executes the code. Again, this is generally only an issue for queued machines. This can be done on a per-code basis for a machine. The default command is:

```
$code $inFile >&! $outFile
```

This can be altered by defining the keyword <Code>Command in the CONFIG.<machine> file where <Code> is NWChem or Gaussian98™. Here is an example using poe for nwchem.

```
NWChemCommand {
poe $nwchem $inFile -procs $nodes >& $outFile
}
```

Depending on how you have compiled NWChem, you may need to modify this section and still make use of NWChem's parallel program.

An example to accomplish this would look like:

```
NWChemCommand {
# - Generating setup file for use with Parallel
if ( -e nwchem.p) rm -f nwchem.p
cat << EOCAT > nwchem.p
$USER `hostname` $totalprocs $nwchem $runDir
EOCAT
<path>/parallel nwchem $inFile >&! $outFile
}
```

Where you supply the full path to parallel.
Note that the commands for this keyword need to be actual csh commands.

Adding a Queue Manager

If Ecce does not currently support the queue system you are using, you can add it yourself assuming it operates similar to the other queuing systems. To do this requires editing of the file <top>/siteconfig/QueueManagers. The steps are:

1. Add your queue system name to the values associated with the QueueManagers keyword within this file. Note that lines ending with '\' are continued on the next line and all entries must be separated by white space.

2. Copy the block of information from one of the existing schedulers such as NQS and create a new block using the exact same name used in the previous step. Note that the ##<keyword>>## should be left as is since values are substituted for these variables at run time. You do not need to be concerned with what these keywords mean.
   1. **submitCommand** - The command used to submit a job to your queue system
   2. **cancelCommand** - The command used to cancel an existing job
   3. **setPriorityCommand** The command used to alter the job priority. For example, on unix systems this would be *renice*. This capability is not currently supported for queued systems so you do not need to fill the value in. However, do not remove this line.
   4. **queryJobCommand** - The queue system command used to check the status of a job.
   5. **queryMachineCommand** - The queue command that displays current system information. The value entered here will be used to support the Ecce Machine browser's "Machine Status" button.
   6. **queryQueueCommand** - The queue command that displays current queue information. The value entered here will be used to support the Ecce Machine browser's "Process Status" button. For example, llq can be used to show LoadLeveler queue information.
   7. **queryDiskUsageCommand** - The queue command that displays current disk usage information. Leaving this as "df -k" or something similar should be appropriate for most installations.
   8. **jobIdParseExpression, jobIdParseLeadingText, jobIdParseTrailingText** - These three fields are used to extract the job id from the output the queue submission commands generate. jobIdParseExpression is a required attribute. It consists of a regular expression for extracting the job id. The jobIdParseLeadingText and jobIdParseTrailingText are optional but useful where regular expressions can't sufficiently extract the job id. If used, they should be used together where the job id is all the text in between what is described by jobIdParseLeadingText and jobIdParseTrailingText. Consider the following example where a job is submitted with llsubmit. The output looks like: ***Job Submitted***
      Logging Job
      llsubmit: Processed command file through Submit Filter: "/usr/local/loadl/sbin/jobfilter".
      llsubmit: The job "sw1515.nwmpp1.emsl.pnl.gov.36465" has been submitted.
      The values of LoadLeveler|jobIdParseExpression: \,.*
      LoadLeveler|jobIdParseLeadingText: \,The job "
   LoadLeveler|jobIdParseTrailingText: \," direct the software to match all the output and then the job id will be taken as everything after *The job"* and before the next ".

3. Modify the eccejobmonitor perl script. This script is located in your <eccetop>/platform/<platform>/bin directory. It runs on the computer server and monitors both the state of the job within the queue system and the output file produced by your job. In order for it to successfully perform its function, it needs to know how to check for job states within your queue system. To complete this step, you need to know the same information as for **queryJobCommand** above, the format of the output of the queryJobCommand and the codes your queue.system uses for various job states. For example, with PBS, 'R" indicates running, while 'QHWTS' are various pending states.

Using your favorite editor, edit the <eccetop>platform<platform>/bin/eccejobmonitor script. Search for “sub JobCheck”. This is the subroutine that you must modify. Since many of the queue systems are derivatives of existing systems, you may be able to readily insert additions into one of the existing if / elsif blocks. For example, since LSF is a derivative of NQS, the proper LSF commands were inserted with the NQS code block. If this is not the case, add a new code block following the “elsif ($q eq ‘globus’)” code block (and before the final else block). You should be able to copy one of the existing blocks of code and modify it for your purposes without too much difficulty. Please read the subroutine header documentation.

4. Use the GUI to configure a machine that uses this queuing system. If you have correctly completed the previous steps, your newly registered queue should now appear in the **Queue Managers** option menu.
5. Complete any customizations required as described previously.

6. Run Ecce to create a calculation and run a test job against the new queue system. Test that you can submit the job, kill a job, and get real-time update of job state. Also run the Ecce machine browser application and ensure that status information can be retrieved and displayed.

Look at the examples for other queuings system for hints on what type of information to provide.

Other Job Script Customization

Other customizations are possible. The most useful of these is to set the environment variables an application may require. For example, NWChem needs to have /usr/ucblib added to the library path on Solaris machines. This can be done with the following key/value assignment:

```
NWChemEnvironment {
 LD_LIBRARY_PATH /usr/ucblib
}
```

This can be put in every Solaris configuration file. In some cases it is more convenient to put this information in the submit.site file.

The following keys are defined:

1. <Code>Environment - e.g., NWChemEnvironment. This block can contain environment variables and their assigned value, one per line.
2. <Code>FilesToRemove - e.g., Gaussian98™FilesToRemove - a list of files that should always be deleted when a job finishes. Typically this means core files or scratch files. Wild cards are supported. You would typically modify this only in the submit.site file.
3. setup - csh code to be inserted into the script prior to the job execution statement. This might be useful for dumping the list of nodes assigned by the queue system for example.
4. wrapup - csh code to be inserted into the job script after the job execution statement.

More Customization

The code that reads the configuration files and generates the job script is a perl script named gensub located in the directory <top>/scripts. You can always customize this if necessary. contact ecce-support@emsl.pnl.gov for assistance.