Chapter 1

Using NQS on a CM-5

NQS stands for *Network Queueing System*, which is a standard batch system. In a batch system, you submit one or more programs as a request to a queue, rather than executing them directly. The batch system in turn submits the requests in the queue for execution. Your request is generally executed when it reaches the head of its queue.

Read this chapter to learn how to use NQS on the CM-5. Read the next chapter if you are a system administrator and want to set up NQS on your CM.

To help you learn how to use NQS, we provide a trivial sample program in the first section of this chapter, followed by instructions on how to compile it. You can submit the compiled program via NQS to be executed on the CM.

1.1 A Simple Program

The program shown below is written in CM Fortran.

The program sets up three arrays of five elements each. The elements of array A are assigned the values 1, 2, 3, 4, 5; the elements of array B are each assigned the value 2. The program then squares each of these values, adds each element of A to the corresponding element of B, and puts the results in array C. It then prints the results. (This, of course, is not a typical data parallel program.)

```
PROGRAM SIMPLE

INTEGER A, B, C, N
PARAMETER (N=5)
DIMENSION A(N) B(N), C(N)

DATA A /1,2,3,4,5/
B=2

C=A**2 + B**2

PRINT */ 'Array C contains:'
PRINT *, C

END
```

Type this program in a file as you normally would; call the file **`simple.fcm`**. (Remember that in Fortran each program statement must begin in column 7.)

To compile the program, issue the following command at your UNIX prompt (which is represented as a percent sign in this manual):

```
% cmf simple.fcm -o simple
```

You now have a CM Fortran program called **`simple`** that is ready for execution on the CM.

1.2 Overview of NQS

As mentioned above, NQS is a standard batch system. NQS can also be used for batch submissions to computers other than the CM. In this manual, however, we focus only on using NQS to submit requests for execution on the Connection Machine system.

The CM system administrator is in charge of configuring NQS queues to meet the needs of your site. A particular partition manager on the CM may not have any queues, or it may have several. You may have only *batch queues*, which submit requests directly to the CM for execution, or you may in addition have *pipe queues*, which pass requests along to batch queues. Pipe queues are useful because they can be associated with several different batch queues; if one is unavailable, the pipe queue can try the next, until it finds one that will accept the request. You don't have to worry about finding the available queue yourself.

Here are some of the characteristics of batch queues that a system administrator can configure:

- *Who can use a queue.* The system administrator can restrict access to a queue to specific users or groups.
- *Limits associated with requests in the queue.* For example, there can be limits on the amount of CPU time or stack size that a request can use.
- *When the queue submits its requests for execution.* A batch queue can operate continuously, or it can operate only at specified times—for example, from midnight to six in the morning.

To submit a request for execution via either a batch queue or a pipe queue, you either:

- Issue the **`qsub`** command, using as an argument the name of a script file that contains the name of the program or programs to be run; or
- Submit the program or programs to **`qsub`** from the standard input

See Section 1.3, below.

Processes in a batch request run under timesharing, just like any other processes.

To obtain information about a queue, or about the status of a request in a queue, issue the **`qstat`** command. See Section 1.5.

Table 1 lists the NQS user commands described in the rest of this chapter. The appendix contains UNIX man pages for all NQS commands.

Table 1. User commands for the NQS batch system.

Command	Meaning
`qdel`	Delete or signal one or more batch requests.
`qlimit`	Display the supported limits on batch queues.
`qstat`	Display the status of queues and batch requests.
`qsub`	Submit a batch request.

1.3 Submitting a Batch Request: The qsub Command

Use the **qsub** command to submit a program for execution via either a batch queue or a pipe queue.

NOTE: The **qsub** command has many options associated with it; in this chapter, we discuss only some of the more important. See the man page for **qsub** in the appendix for a complete discussion of all its options. Table 2 summarizes the **qsub** options supported on the CM-5.

Table 2. Options for the qsub command.

Option	Meaning
-a *time*	Do not run the request before the specified time and/or date.
-e *filename*	Direct the standard error output to the specified file.
-eo	Direct the standard error output to the batch request output file.
-ke	Keep the standard error output on the execution machine.
-ko	Keep the standard output on the execution machine.
-lc *size*	Set the per-process corefile size limit.
-ld *size*	Set the per-process data-segment size limits.
-lf *size*	Set the per-process permanent-file size limit.
-ln *value*	Set the per-process nice execution value limit.
-ls *size*	Set the per-process stack-segment size limit.
-lt *time*	Set the per-process CPU time limit.
-lw *size*	Set the per-process working set limit.
-mb	Send mail when the request begins execution.
-me	Send mail when the request ends execution.
-mr	Send mail when the request restarts.
-mt	Send mail when the request is transported between queues.
-mu *username*	Send mail about the request to the specified user.
-nc	Declare that the request is not recoverable.
-nr	Declare that the request is not restartable.
-o *filename*	Send the output of the request to the specified file.
-p *priority*	Set the priority for the request in the batch queue.
-q *queue*	Send the request to the specified batch queue.
-r *name*	Assign the specified request name to the request.
-re	Remotely access the standard error output file.
-ro	Remotely access the standard output file.
-s *shell*	Use the specified shell to interpret the request.
-x	Export all environment variables with the request.
-z	Submit the request silently.

1.3.1 The Basics

To execute the program **simple** via the queue **cmq1**, put the program's name in a *script file*. A script file is simply a UNIX file that contains commands to be executed. For example, you could create a file that contains just the word **simple**; you could call the file **/yourname/simple_script**. You could then submit this file to **cmq1** as follows:

```
% qsub -q cmq1 /yourname/simple_script
```

The **-q** flag specifies the name of the queue to which you are submitting the request.

The system displays a response like this:

```
Request 276.barney.acme.com submitted to queue: cmq1
```

The number 276 is a *sequence number* assigned to this request by NQS, and **barney.acme.com** is the *hostname* of the computer from which the request was submitted; **276.barney.acme.com** is the *request-id* for this request.

When **simple** is finally executed, its output is placed in a file; error messages are placed in another file.

Submitting a batch request has these basic elements:

- Specifying the queue to which the request is being submitted
- Specifying the request to be run
- Specifying options that affect the way the request is to be run

You can embed **qsub** options at the beginning of the script file, along with the name of the executable program and other commands. See Section 1.3.2, below, for an example. NQS looks at options in the script file only if they are not specified on the **qsub** command line; this lets you override a script file option by specifying a different setting for the option on the **qsub** command line.

1.3.2 Specifying the Queue

There are several different methods of specifying the queue to which you want to submit your request. You can find out the names and characteristics of available batch queues or pipe queues by issuing the **qstat** command with the **-b** or **-p** option, respectively; see Section 1.5.2.

You can use the following methods to specify a batch queue:

- Use the **-q** option on the **qsub** command line, as shown in the example above. NQS submits the request to the queue you specify.

- Embed the **-q** option in a script file that you name on the **qsub** command line. All **qsub** options must appear at the beginning of the script file, and must begin with a pound sign (#) followed by an "at" sign (@) and a dollar sign ($). The option must begin immediately after the dollar sign—no white space is allowed. Comments must begin with a pound sign. For example, the following script file sends the program **simple** to queue **cmq1** for execution:

```
#
# Example of a batch script file
#
# @$-q cmq1 # Send request to cmq1 unless
#           # overridden on command line.
#
simple
```

  If you named this script file **simple_script**, you could execute the program by issuing the following command:

```
% qsub simple_script
```

- Set the environment variable **QSUB_QUEUE** to the name of the queue to which you want the request submitted. You would typically do this to set up a default queue for all requests, which you could override for a specific request by using the **-q** option. If you use the C shell, you could put the following command in your **.cshrc** file to set the default queue to **cmq2**:

```
setenv QSUB_QUEUE cmq2
```

If you don't use any of these methods for specifying a queue, the request is submitted to the default batch queue for the system, if your system administrator has defined one.

1.3.3 Specifying a Request from a Script File

As we have already shown, you can execute a program by including its name in a script file. You can include UNIX commands in the file as well. Typically, NQS interprets the commands in a script file exactly as if you had typed them at your UNIX prompt. It may, however, use a different shell to interpret the commands, depending on how your system administrator has configured NQS. See Section 1.3.7, below.

1.3.4 Specifying a Request from Standard Input

Instead of using a script file, you can simply enter the request from standard input—that is, directly after the **qsub** command line. Put each command or program name on a separate line, and type the **Ctrl-d** key combination at the end to signal that there is no more input. For example:

```
% qsub -q cmq1
  simple
  cmsetsafety on
  simple
  Ctrl-d
```

If you are executing a shell under Emacs or Gmacs, type **Ctrl-c Ctrl-d**.

1.3.5 The Output from a Request

NQS places the output from a batch request in a file, which is by default placed in the directory from which you issued **qsub**. You can control the name and location of this file. The default filename consists of the first seven characters of the script name, followed by **.o**, followed by the sequence number of the request. Thus, in our example in Section 1.3.1, NQS would put the output in the file **sim-**

ple_.o276 in your current working directory. Messages to standard error go into a file with **.e** in the name instead of **.o** (**simple_.e276** in this example).

If you submit the request from standard input, the default output and error files would begin with **STDIN.o** and **STDIN.e**, followed by the sequence number.

To specify a different output filename, use the **-o** option, followed by a pathname, on the **qsub** command line or in a script file. NQS writes output to the pathname you specify. For example,

```
% qsub -o /requests/simple.out simple
```

causes the standard output of the program **simple** to be written to **/requests/simple.out**.

Similarly, use the **-e** option to specify a different pathname for standard error output.

Another way to change the name of the output file is to use the **-r** option, followed by a request-name of up to 15 characters. This request-name identifies the request when you issue the **qstat** command to check the status of requests; if you don't specify a request-name, NQS uses the name of the script file (or **STDIN**) instead. If you do specify a request-name, NQS substitutes it for the name of the script file (or **STDIN**) in the name of the output file.

Here is sample output for our **simple_script** batch request:

```
Cold boot...
Array C contains:
            5      8      13     20     29
FORTRAN STOP
logout
```

NOTE: In processing the batch request, NQS runs a script as if it were logged in as you; start-up files like **.cshrc** and **.login** are executed. This means that you may see various messages along with the output. In particular, you will probably see the following message:

```
Warning: no access to tty; thus no job control in this
shell...
```

This comes from the shell, warning you that there is no terminal associated with this job. You can ignore this message.

NOTE: While the request is running, NQS considers your home directory to be your current working directory (because it runs the request as a newly logged-in process). Thus, if the process dumps core, the corefile is placed in your home directory, rather than the directory from which you submitted the request.

1.3.6 Setting Limits on a Request

As shown in Table 2, the **qsub** command has many options you can specify to set limits on the amount of resources a batch request can use. The batch queue has its own set of limits. You can find them out by issuing the **qstat** command with the **-l** option; see Section 1.5.2. You may want to set lower limits to obtain more favorable scheduling for your request, or to avoid running up accounting charges if, for example, your program goes into an infinite loop. If the limit you set *exceeds* the limit for a queue, the queue will not accept the request.

For example, use the **-lt** option to set a limit on the amount of partition-manager CPU time an individual program within a batch request can use. The following command sets a limit of 120 seconds of CPU time for the program **simple**:

```
% qsub -lt 120 simple
```

CMOST does not support all of the limit options that **qsub** lets you specify. If you specify an unsupported option, NQS ignores it. To find out which options your site supports, issue the **qlimit** command. For example, if your partition manager is named Barney, issue **qlimit** as follows:

```
% qlimit barney
```

The response might look like this:

```
Core file size limit (-lc)
Data segment size limit (-ld)
Per-process permanent file size limit (-lf)
Nice value (-ln)
Stack segment size limit (-ls)
Per-process cpu time limit (-lt)
Working set limit (-lw)

Shell strategy = FREE
```

These are the limits you can set for this partition manager. The “shell strategy” in this response refers to the default way in which NQS chooses a shell to interpret commands in a script file. See Section 1.3.7, below.

NOTE: Other limits may be available if you are sending your request to another computer.

1.3.7 Choosing a Shell

As we mentioned above, your system administrator can specify how NQS is to interpret commands in batch script files. This is called the *shell strategy*; you can find out the default shell strategy via the **qlimit** command. The possible shell strategies are:

- *Free*. Your login shell determines the appropriate shell to be used to execute the commands in your script file, and executes that shell. This typically means that NQS uses the shell that would have been used if you had issued the commands in the script file interactively. For example, if your script file begins with the line

  ```
  #! /bin/csh
  ```

  your login shell would execute a C shell for the script file.

- *Login*. NQS uses your login shell to execute the commands in your script file, regardless of the contents of your file.

- *Fixed*. NQS uses a specified shell to execute the commands, regardless of the contents of your script file. Use **`qlimit`** to find out the name of this shell.

You can override the shell strategy by using the **`-s`** option with the **`qsub`** command. For example,

```
% qsub -s /bin/csh simple_script
```

specifies that the C shell is to be used to interpret the commands in the script file **`simple_script`**.

1.3.8 Setting a Priority for a Batch Request

To set a priority for your batch request in its queue, use the **`qsub`** option **`-p`**, followed by an integer from 0 to 63, inclusive; 63 is the highest priority, and 0 is the lowest priority. This priority determines the request's position in the queue. The request is placed in front of all requests with lower priority, and behind all requests with higher or the same priority.

If you don't specify a priority, the request is assigned a default priority, as set by the system administrator. Use the **`qstat`** command with the **`-fb`** or **`-fp`** option to determine the default priority for a queue; see Section 1.5.2.

NOTE: NQS does not necessarily run requests in the order in which they appear in a batch queue. It can take requests out of order to use resources efficiently. Generally, however, requests at the beginning of the queue are run before requests that appear later in the queue.

1.3.9 Receiving Mail about a Batch Request

Use the **`qsub`** options **`-mb`** and **`-me`** to specify that NQS is to send you mail about your batch request. Specify **`-mb`** to get mail when the request begins execution; specify **`-me`** to get mail when the request ends execution. Use the **`-mu`** option with a username to send the mail to another user.

To obtain more information about the status of a batch request, use the **`qstat`** command with the **`-a`** option; see Section 1.5.1.

1.4 Deleting a Batch Request: The qdel Command

Issue the **qdel** command to delete a request from a queue. As an argument, specify the request-id that was displayed when you submitted the request. (You can also obtain the request-id by issuing the **qstat** command with the **-a** option; see Section 1.5.1, below.) For example,

```
% qsub -q cmq1 simple_script
Request 276.barney.acme.com submitted to queue:
cmq1
% qdel 276
```

submits a request, then deletes it. (You don't need to specify the hostname if you are issuing the command from the local host.)

This form of the **qdel** command does not delete a request that is actually running. To do this, use the **-k** option. This option sends a SIGKILL signal to the specified request, causing it to exit and be deleted from the queue. If the request contains more than one process, all are signalled.

To send a signal other than SIGKILL to a running request, specify its number instead of **k** (see the discussion of **sigvec** in your UNIX documentation for signal numbers). For example, to send a SIGTERM signal to a running request with request-id 276, issue this command:

```
% qdel -15 276
```

1.5 Obtaining Information: The qstat Command

Use the **qstat** command to obtain information about queues and batch requests. This section describes some of the more important options; for complete information on **qstat**, see its man page in the appendix.

1.5.1 Finding Out about Batch Requests

If you issue **qstat** with no options, the status of all your requests on the local host are displayed. To find out the status of all requests on any queue, issue **qstat** with the **-a** option. To find out the status of the requests of a particular user, use the **-u** option, followed by the user's name. For example:

```
% qstat -u sharon
```

The response might look like this:

```
================================================================
NQS Version: 2        REQUESTS on diamond.think.com
================================================================
REQUEST      NAME   OWNER   QUEUE    PRI  NICE   CPU    MEM       STATE
0@diamond.  cm5sub sharon  pn_128   32     20  36000  UNLIMITED  RUNNING
```

This shows that user Sharon has one request, with request-id **0@diamond.think.com** and name **cm5sub**, running on the queue **pn_128**. The other fields are:

- **PRI** — This is the intraqueue priority for the request; see Section 1.3.8.
- **NICE** — This is the nice value for the request. If you don't specify this value via the **-n** option to the **qsub** command, NQS uses the default value for the queue. (See your UNIX documentation for more information on nice values.)
- **CPU** — This is the maximum partition-manager CPU time for the request; once again, you can set this via a **qsub** option (**-lt**), or you can accept the default queue limit.
- **MEM** — This is the maximum memory for the request; for requests running on a CM-5, this will always be UNLIMITED.
- **STATE** — This is the state of the request. A request can be in one of these states:
  - **Arriving** — A request is *arriving* if it is being placed on the queue from a remote host.
  - **Holding** — A request is *holding* if it is currently prevented from entering any other state because a hold has been placed on it.

- **`Waiting`** — A request is *waiting* if it was submitted with the constraint that it not be run before a certain date or time, and that date or time hasn't arrived yet. You submit a request in this way by using the **`qsub`** option **`-a`**.
- **`Queued`** — When a request is *queued*, it is eligible to run as soon as the resource is available or the queue's restriction window opens.
- **`Routing`**, **`Departing`** — If the queue is a pipe queue, a request can be *routing* or *departing* as it passes through the queue.
- **`Staging`** — A request is *staging* when its input files are being brought on to the partition manager on which it is to execute.
- **`Running`** — A request is *running* when it is actually executing.
- **`Exiting`** — A request is *exiting* when it has completed execution and the required output files are being returned.

1.5.2 Finding Out about Queues

Use **`qstat`** with the **`-b`** option to find out the status of NQS batch queues. The response might look like this:

```
========================================================
NQS Version: 2   BATCH QUEUES on diamond.think.com
========================================================
QUEUE NAME     STATUS   TOTAL   RUNNING  QUEUED  HELD TRANSITION
---------------------------------------------------------------------
q1             AVAILBL    1       1/1       0      0       0
q2             AVAILBL    2       2/10      0      0       0
```

There are two batch queues, **`q1`** and **`q2`**, on **`diamond.think.com`**. The other fields provide the following status information about the queues:

- **`STATUS`** — A queue can be in one of these states:
  - **`AVAILABL`** — The queue is *enabled* (that is, you can submit requests to it) and *started* (that is, submitted requests can run).
  - **`STOPPED`** — The queue is enabled but stopped (that is, the queue is enabled, but queued requests are blocked from running).

  - **`DISABLED`** — You cannot submit requests to the queue.
  - **`UNAVAIL`** — The queue is disabled and stopped.
  - **`NQS DOWN`** — The NQS daemon for the partition manager is not running.

  You can submit requests to a queue only if the queue is enabled and the local NQS daemon is present.

- **`TOTAL`** — This is the number of requests currently in the queue.
- **`RUNNING`** — The first value in this field is the number of requests that are running in the queue. The second value is the *run limit* for the queue (that is, the maximum number of requests that can be running in the queue at the same time). Thus, in the example, **`q1`** has one request running, and its run limit is 1; **`q2`** has two requests running out of a possible 10.
- **`QUEUED`** — This is the number of requests in the queue waiting to run.
- **`HELD`** — This is the number of requests that are ineligible for processing.
- **`TRANSITION`** — This is the number of requests that are in between queues (for example, between a pipe queue and a batch queue).

Use the **`-p`** option to find out this information about pipe queues.

Use the **`-f`** option along with **`-b`** or **`-p`** to display information about a queue in a fuller format. For example,

```
% qstat -fb bq0
```

might produce this response about the batch queue **`bq0`**:

```
========================================================
NQS version:2  BATCH      QUEUE:  bq0.eno.think.com    status:  AVAILBL
=========================================================
                                        Priority: 32
ENTRIES:
        Total:   0      Running: 0
        Queued:  0      Held:    0      Transiting: 0
COMPLEX MEMBERSHIP:

RESOURCES:
  Per-proc core file size limit= 1 megabytes <DEFAULT>
  Per-process data size limit  = 1 megabytes <DEFAULT>
  Per-proc perm file size limit= 1 megabytes <DEFAULT>
  Per-proc execution nice value= 0 <DEFAULT>
  Per-process stack size limit = 1 megabytes <DEFAULT>
  Per-process CPU time limit   = 36000.0 <DEFAULT>
  Per-process working set limit= 1 megabytes <DEFAULT>
  Restriction Window start time = <NONE>
  Restriction Window stop time = <NONE>
  Restriction Window mode = <NOT SET>
  Restriction Window send terminate signal = <NOT SET>

ACCESS
  Unrestricted access
```

Comments about some of these fields:

- The **`COMPLEX MEMBERSHIP`** field indicates whether or not this queue belongs to a *queue complex*. Multiple queues can belong to a queue complex that shares a single run limit.
- The **`Restriction Window`** fields indicate the time during which the queue is available to run requests.
- The **`ACCESS`** field indicates what access restrictions, if any, have been placed on the queue. The system administrator can restrict access to specified users and groups.