Appendix A

Global Configuration File

This appendix presents a sample global configuration file, and then describes each of the fields in this file. Comments in parentheses explain certain values; they are not actually part of the global state file.

In general, time-related fields are measured in seconds and memory-size fields are measured in Kbytes. A value of 0 (zero) for a resource generally means that the value is unlimited. (Fields for which a value of zero denies access are identified in the field descriptions.)

```
nextjid  8168
nextgid  31658
prompt_cputime300                       (5 minutes)
prompt_mem_per_proc 4096                (4 megabytes)
prompt_nproc  64
global_user_nsub_limit  4
global_user_nrun_limit  4
global_part_nrun_limit  5
global_int_part_nrun_limit   6
global_user_part_nrun_limit  4
global_job_cputime_limit     86400      (1 day)
always_believe_requested_memory   no
default_pri   50
default_sp_memory  0
default_sp_disk    0
default_safety 0
foreign_requested_cputime    86400      (1 day)
foreign_requested_mem_per_proc    0     (unlimited)
foreign_requested_total_mem  4194304    (4 gigabytes)
foreign_hard_cputime    900             (15 minutes)
foreign_hard_mem_per_proc    0          (unlimited)
foreign_hard_total_mem  4194304         (4 gigabytes)
idle_timelimit 3600
```


Copyright © 1993 Thinking Machines Corporation

```
dedicated_grace_period  600            (10 minutes)
dedicated_submit_by_limit 14400        (4 hours)
maximum_future_reconfig  259200        (3 days)
maximum_memory_fraction_for_draining 0.75
operators 8 : 0 2014 2223 2185 78 2023 2230 8733
managers 2 : 0 2014
```

nextjid, **nextgid**
: The job ID and group ID numbers that will be assigned to the next submitted job.

prompt_{cputime | mem_per_proc | nproc}
: Determines the default answers for the resource questions asked by **jrun** and **jsub**.

global_user_nsub_limit
: The maximum number of jobs that a single user may submit concurrently.

global_user_nrun_limit
: The maximum number of jobs that a single user may run concurrently.

global_part_nrun_limit
: The maximum number of jobs that may run concurrently in one partition. This limit is applied to non-interactive jobs (those run via **jsub**).

global_int_part_nrun_limit
: The maximum number of jobs that may run concurrently in one partition. This limit is applied to interactive jobs (those run via **jrun**). It is typically higher than **global_part_nrun_limit**.

global_user_part_nrun_limit
: The maximum number of jobs that any one user may run concurrently in one partition.

global_job_cputime_limit
: The maximum CPU time that can be requested when submitting a job.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

always_believe_requested_memory
: Is **yes** if the scheduler should always allocate the requested memory for a job, even if the job actually uses less memory. By default (**no**), the scheduler will give unused requested memory back to the system 2 minutes after the job begins.

default_{pri | sp_memory | sp_disk | safety}
: The default values for attributes not specified by the user. See the **priority**, **sp_memory**, **sp_disk**, and **sp_safety** options to **jsub** for an explanation of these values.

foreign_requested_cputime
: The implicit soft time limit (in seconds) for foreign jobs. This should be kept small. Jobs that exceed this limit have their score decremented, allowing them to be killed if queued jobs want to run.

foreign_requested_mem_per_proc
: The implicit soft memory limit, measured in kilobytes per processor, for foreign jobs. Jobs that exceed this limit have their score decremented, allowing them to be killed if queued jobs want to run.

foreign_requested_total_mem
: An alternative way of setting the soft memory limit, measured in total kilobytes of memory.

foreign_hard_cputime
: The implicit hard time limit for foreign jobs in seconds. Jobs that exceed this limit are killed immediately.

foreign_hard_memory
: The implicit hard memory limit for foreign jobs measured in kilobytes of memory per processor. Jobs that exceed this limit are killed immediately.

foreign_hard_total_mem
: An alternative way of setting the hard memory limit, measured in total kilobytes of memory.

Copyright © 1993 Thinking Machines Corporation

`idle_timelimit`

Sets a timeout limit for idle jobs. Jobs that are idle for more that the specified period of time are killed.

`dedicated_grace_period`

Defines the limit (amount of time after the last dedicated job runs) after which a volatile dedicated partition will revert to normal mode.

`dedicated_submit_by_limit`

Sets a deadline for the submission of dedicated jobs. It defines the minimum time interval, measured in seconds, between the time when a user submits a job and the beginning of a dedicated execution period. If the user misses the deadline and no other dedicated jobs are queued, the dedicated execution period is cancelled, unless the **`-lock`** option is supplied to the reconfiguration command.

`maximum_future_reconfig`

Sets how far in advance a future reconfiguration event can be scheduled. Future reconfiguration events that are more than *n* seconds in advance are ignored. This allows you to create scripts with future reconfiguration events that are months in advance, without burdening the daemon with lots of CPU overhead in order to scan them all. The default is 3 days (259200 seconds). Scheduled future reconfiguration events are usually put into a shell script that can be run on a regular basis from **`cron`**.

`maximum_memory_fraction_for_draining`

Sets the maximum amount of memory that a user can request, expressed as a fraction of the total physical memory of a partition, under which draining will occur. The default is 0.75 (75 percent). Jobs that request more than this amount of memory ought to be run in dedicated mode.

Copyright © 1993 Thinking Machines Corporation

`operators`, **`managers`**

Lists of user IDs validated to perform those functions. Operators can issue **`jop`** commands. Managers can issue **`mgr`** commands. The format of the field is:

*field N : val [val *]*

where *N* is the number of values that will be supplied after the colon (:).

Copyright © 1993 Thinking Machines Corporation

Appendix B

Queue Configuration File

This appendix presents an example of the configuration file for the default queue, then discusses the fields and their values.

```
name default
search_order  100
enabled       yes
started       yes
users 0 :
groups 0 :
projects 0 :
exact_mem_users 2 : 0 3453
exact_mem_groups 0 :
exact_mem_projects 0 :
max_memory    0
max_cputime   0
max_cpu_times_mem_per_proc   0
max_nproc     0
min_memory    0
min_cputime   0
min_cpu_times_mem_per_proc   0
min_nproc     0
min_pri  0
max_pri  100
foreign  no
range_baseproc -1
range_nproc   0
dedicated     no
user_nsub_limit    999
user_nrun_limit    999
user_part_nrun_limit    999
queue_nrun_limit   999
queue_part_nrun_limit   999
```


Copyright © 1993 Thinking Machines Corporation

```
queue_mem_run_limit 0
queue_part_mem_run_limit     0
queued_base_factor 100000
queued_wait_factor 1
queued_nproc_factor-0.25
queued_pri_factor  0
queued_cputime_factor   0
queued_memory_factor    0
queued_mem_cpu_product_factor 0
queued_already_queued_factor -3600
running_base_factor 1000000
running_nproc_factor    0
running_memory_factor   0
running_cputime_exceeded_factor   -2000000
running_memory_exceeded_factor    -2000000
running_mem_cpu_product_exceeded_factor     -2000000
running_interactive_factor   1000
running_dedicated_late_factor     -2000000
```

name Specifies the name of the queue.

enable Updated by **djm_master**. States whether the queue is available for accepting jobs. It does not mean that jobs will run. The queue must be *started*, as well as *enabled*, to be able to schedule jobs.

users, **groups**, **projects**
Specify the user, group, and project IDs that are allowed to submit jobs to the queue. As with the **operators** and **managers** fields in the global state file, the value preceeding the colon (:) gives the number of values that are supplied after the colon.

exact_mem_{users | groups | projects}
Specify the user, group, and project IDs that are allowed to request exact memory. Format is the same as for the preceding fields: number of entries followed by a colon (:) followed by the entries themselves.

max_{memory | cputime | nproc | cpu_times_mem_per_proc}
These fields are the per-request maximum values for memory, CPU time, number of processors, and the product of CPU time and memory. A value of 0 means that the field is unlimited. Note that these fields

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

should not be used for the foreign queue, as the **foreign_requested_{total_mem | mem_per_proc | rcputime}** in the global configuration file (see Appendix A) specifies the values for the foreign queue.

min_{memory | cputime | nproc | cpu_times_mem_per_proc}
These fields specify the minumum values for these options. Here, a value of 0 means just that.

{min|max}_pri Specifies the minimum and maximum priorities the queue will accept for the **-priority** switch to **jsub** and **jrun**. Priority is used to determine a job's score (see Appendix C).

foreign States whether the queue will accept foreign jobs. The value can be **yes** or **no**. In normal operations, there should only be one queue that can accept foreign jobs. All other queues should have a **no** value for this field.

range_baseproc, **range_nproc**
Specify the processors upon which the queue operates. Normally, **range_baseproc** is **-1** and **range_nproc** is **0**. This allows a queue to operate on all the processors within a DJM partition. To restrict queued jobs to operating on a subset of the processors, set **range_baseproc** to the lowest-numbered processor in the range and set **range_nproc** to the number of processors on which the queue is to operate.

dedicated States whether the queue is to used be used for running jobs in dedicated mode. See Section 2.2.1 for a discussion on running DJM in dedicated mode. The value for this field is either **yes** or **no**.

user_nsub_limit, **user_nrun_limit**
Specify the number of jobs that a single user may submit or have running (respectively) at a given time. In these fields, a value of **0** means just that.

user_part_nrun_limit
Defines the maximum number of jobs a single user may have running in a single partition at any given time.

Copyright © 1993 Thinking Machines Corporation

`queue_nrun_limit`
: Specifies the maximum number of jobs that can be running in the queue at once. Further job submissions are queued in the state "**`Que unj`**", even if resources are available to run the submitted job.

`queue_part_nrun_limit`
: Defines the maximum number of jobs that the queue can run in a partition at any given time.

`queue_mem_run_limit`
: Specifies the maximum amount of memory that the entire queue may use at any given time. Further job submissions to the queue are queued in the "**`Que qnm`**" state until the total is low enough to support the job. A 0 represents an unlimited value.

The rest of the fields in the queue file are used in the computation of a job's score (used for scheduling jobs). Refer to Appendix C for a discussion of these fields and scoring in general.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

Appendix C

Score Equations

C.1 Score Equations for Queued Jobs

The score is calculated as shown below. Parameters within angle brackets (< >) are derived from the queue's configuration file, shown in Appendix B.

Scores are integer values in the range of 1 to 999,999,999. The constants used to specify parameters within files can be either positive or negative integers; the only limit is that the numbers you use must keep the scores within their allowed limits. (For example, negative integers cannot overbalance positive ones so strongly as to turn a score negative.)

When you first set up DJM, you should use the **`jstat scores`** command to monitor the scores for running and queued jobs. You can then use this information to determine the correct queued and running factors for each queue to assure a fair allocation of resources.

```
<queued_base_factor>
   + (<queued_wait_factor> * number of seconds queued)
   + (<queued_nproc_factor> * number of processors requested)
   + (<queued_memory_factor> * kilobytes of memory requested)
   + (<queued_cputime_factor> * number of CPU seconds requested)
   + (<queued_mem_cpu_prod_factor> * kilobytes of memory requested
           * number of CPU seconds requested)
   + (<queued_already_queued_factor> * number of jobs already queued)
   + (<queued_pri_factor> * user job priority)
```


Copyright © 1993 Thinking Machines Corporation

C.1.1 Queued Score Fields in the Queue Configuration File

The **queued_base_factor** is the constant score that all jobs start with when submitted. The score is then modified as indicated by the formula. The value should be tuned to be commensurate with the value used in the **running_base_factor** so that all queued jobs eventually run.

The **queued_wait_factor** field is the value to be added to the score for each second the job is in the queue. Note that by setting this value to a number greater than 1, you can get a superlinear increase in the score over time.

The **queued_nproc_factor** increases the score in proportion to the number of processors requested. This is usually either zero or marginally negative; the latter penalizes those who request large numbers of processors.

The **queued_memory_factor** increases the score in proportion to the amount of memory requested. This too is usually either zero or marginally negative; the latter penalizes those who request large chunks of memory.

The **queued_cputime_factor** increases the score in proportion to the amount of CPU time requested. This too is usually either zero or marginally negative; this penalizes those who request large chunks of time.

The **queued_pri_factor** is used to proportion the effect of the user's desired priority. If the user specifies no **-priority** switch, then the default value from the global state file is used for the user's priority, multiplied by the **queued_pri_factor**, and added to the score.

The **queued_already_queued_factor** increases the score in proportion to the number of jobs already queued by the user. It is usually negative, to prevent a single user from hogging the machine by submitting a large number of jobs.

The **queued_mem_cpu_product_factor** increases the score in proportion to the product of the requested memory and CPU time. It is usually zero or negative. It can be used in conjunction with **queued_cputime_factor** and **queued_memory_factor** to adjust job scores.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

C.2 Score Equation for Running Jobs

The score for running jobs is calculated as shown below. Parameters within angle brackets (**< >**) are derived from the queue's configuration file, shown in Appendix B.

Again, scores are integer values in the range of 1 to 999,999,999. The constants used within files can be either positive or negative integers; the only limit is that the numbers you use must keep the scores within their allowed limits. (For example, negative integers cannot overbalance positive ones so strongly as to turn a score negative.)

```
<running_base_factor>
    + (<running_nproc_factor> * number of processors job is using)
    + (<running_memory_factor> * kilobytes of memory job is using)
    + <running_cputime_exceeded_factor>
    + <running_memory_exceeded_factor>
    + <running_mem_cpu_prod_exceeded_factor>
    + <running_interactive_factor>
    + <running_dedicated_late_factor>
```

C.2.1 Running Score Fields in Queue Configuration File

The **running_base_factor** is a constant that all running jobs start with. It should be a large value, larger than the score of any queued job under normal conditions. This prevents a queued job from having a higher score than a running job that hasn't overrun any of its requested parameters.

The **running_nproc_factor** increases the score in proportion to the number of processors the job is using. This is usually zero.

The **running_memory_factor** increases the score in proportion to the amount of memory the job is using. This is usually zero.

The **running_cputime_exceeded_factor** is added to the score when a job exceeds its requested CPU time or the partition's soft CPU time limit. This should be a large negative number.

Copyright © 1993 Thinking Machines Corporation

The **`running_memory_exceeded_factor`** is added to the score when a job exceeds its requested memory allocation or the partition's soft memory limit. This should be a large negative number.

The **`running_mem_cpu_prod_exceeded_factor`** is added to the score if the job exceeds the partition's soft limit on the product of the memory times CPU seconds (i.e., kilobyte-seconds). It is usually a large negative value.

The **`running_interactive_factor`** is added to the score for foreign jobs. This can be used to control the ability of the system to run foreign jobs. The value is usually zero. Note that foreign jobs must be able to run for at least 2 minutes (120 seconds), or swapped CM jobs will be killed.

The **`running_dedicated_late_factor`** is added to the job's score if the job is dedicated and has overrun the dedicated time slot. This allows dedicated jobs to continue to run after the system switches to normal mode if there are no other jobs waiting to run. The value is usually a large negative number.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

Appendix D

Partition Configuration File

This appendix presents an example of the partition configuration file, followed by a description of the entries. Unless otherwise noted, **`0`** means unlimited. This file is modified by **`djm_master`** using the reconfiguration commands executed by the operator. Items in angle brackets are optional.

```
current_partition:
current_mode   normal
current_nproc  64
current_baseproc    0
current_reconfig_time    745010133
current_want_timeshared  yes
current_lock   no
current_server_name mars
current_hard_cputime 0
current_hard_mem_per_proc 0
current_hard_cpu_times_mem_per_proc     0
current_soft_cputime 0
current_soft_mem_per_proc 0
current_soft_cpu_times_mem_per_proc     0
<current_queue_argv           queue>
current_total_run_limit  0
<current_offer_argv           resource>
<current_need_argv            resource>

future_partition:
future_fid 1
future_mode    dedicated
future_nproc   64
future_baseproc 0
future_want_timeshared   yes
future_lock    no
future_unlock  no
```


Copyright © 1993 Thinking Machines Corporation

```
future_server_name  mars
future_reconfig_time 745010133
future_reconfig_kill no
future_nwarnings    0
future_hard_cputime 0
future_hard_mem_per_proc 0
future_hard_cpu_times_mem_per_proc 0
future_soft_cputime 0
future_soft_mem_per_proc 0
future_soft_cpu_times_mem_per_proc 0
<future_queue_argv            queue>
future_total_run_limit   0
<future_offer_argv            resource>
<future_need_argv             resource>
```

D.1 Defining the Current Partition

The first set of attributes in the file defines the current partition:

current_mode The execution mode: normal, dedicated, or shutdown.

current_nproc The number of processors in the partition.

current_baseproc
The physical number of the first processor in the partition.

current_reconfig_time
The time at which the partition became current, measured in seconds since Jan 1 1970.

current_want_timeshared
yes if a timesharing daemon is desired. This is always **yes** on a CM-5.

current_lock **yes** if the partition is locked. This is used to prevent future partitions from taking effect (e.g., to make dedicated mode non-volatile).

current_server_name
The name of the server for this partition.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

current_hard_cputime

The hard CPU time limit for jobs run in this partition, measured in seconds. Queued jobs that request more than this amount of CPU time are not scheduled to run in this partition. Running jobs that consume more than this amount of time are killed.

current_hard_mem_per_proc

The hard memory limit for jobs run in this partition, measured in kilobytes per processor. Queued jobs that request more than this amount of memory are not scheduled to run in this partition. Running jobs that consume more than this amount of memory are killed.

current_hard_cpu_times_mem_per_proc

The limit for the product of CPU time and memory, measured in kilobyte-seconds per processor. Queued jobs that request more than this amount are not scheduled to run in this parition. Running jobs that consume more than this amount are killed.

current_soft_cputime

The soft CPU time limit. Queued jobs requesting more than this amount of time will not be scheduled to run in this partition unless the job is **-interruptable**. Running jobs that exceed this amount will suffer a reduction in their job score (see **running_cputime_exceeded_factor**), possibly making them vulnerable to being killed if other jobs are waiting to run.

current_soft_mem_per_proc

The soft memory limit, measured in kilobytes per processor. Queued jobs requesting more than this amount of memory will not be allowed to run unless the job is **-interruptable**. Running jobs that exceed this amount will suffer a reduction in their job score (see **running_memory_exceeded_factor**), possibly making them vulnerable to being killed if other jobs are waiting to run.

current_soft_cpu_times_mem_per_proc

The soft limit on the product of CPU times memory,

Copyright © 1993 Thinking Machines Corporation

measured in kilobyte-seconds. Queued jobs requesting more than this amount will not be allowed to run unless the job is **-interruptable**. Running jobs that exceed this amount will suffer a reduction in their job score (see **running_mem_cpu_product_exceeded_factor**), possibly making them vulnerable to being killed if other jobs are waiting to run.

current_queue_argv
A list of valid queues, one per line. A job is not scheduled to run on the partition unless it is located in one of the queues listed here. If no queues are listed, jobs are allowed from any queue.

current_total_run_limit
The maximum total number of jobs that may concurrently run on this partition.

current_offer_argv
A list of offered resources, one per line. A job is not scheduled to run on the partition unless the partition offers at least the resources required by the user (via **-want**).

current_need_argv
A list of needed resources, one per line. A job is not scheduled to run on the partition unless the user specifies all the resources listed. (This prevents users from tying up resources they do not need.)

D.2 Defining the Future Partition

The remaining attributes in the partition configuration file define a future partition:

fid A unique identifier, used by the **wipe_reconfig** command.

future_mode The execution mode: normal, dedicated, or shutdown.

future_nproc The number of processors in the partition.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

`future_baseproc`
The physical number of the first processor in the partition.

`future_want_timeshared`
`yes` if a timesharing daemon is desired. This is always **`yes`** on a CM-5.

`future_unlock` Reverses the effect of the **`current_lock`** attribute. This reenables the processing of reconfigurations.

`future_server_name`
The name of the server for this partition.

`future_reconfig_time`
The time at which reconfiguration will occur, measured in seconds since 00:00:00 Jan 1 1970 GMT.

`future_reconfig_kill`
`yes` if old jobs should be killed when reconfiguration ocurs.

`future_nwarnings`
The number of warning messages that have been issued to jobs.

The other parameters are the same as for the current partitions.

Copyright © 1993 Thinking Machines Corporation

Appendix E

Sample Job File

This appendix presents and then discusses a sample job file.

```
% cat old/4
jid     4
uid     1584
sid     5580
state   Dead
jobname router_m
queue_name      default
uname   newton
file    /os/cmost-73.4/usr/diag/tsd/router_test
cdir    /users/newton/tmp
stdout_name     router_test.o4
term_name       xterm
ttyname /dev/ttyp3
pri     50
when_queued     745011512
when_run        745011554
active_pid      31063
accum_cputime   10
active_when_started     745011581
max_memory      1296
is_timeshared   yes
actual_nproc    4
actual_baseproc 8
requested_cputime       300
requested_memory        4096
requested_nproc 4
requested_baseproc      -1
dedicated       no
rerun   0
sp_disk 0
```


Copyright © 1993 Thinking Machines Corporation

```
sp_memory       0
server_name     mars.think.com
server_interface        0
argv    /os/cmost-73.4/usr/diag/tsd/router_test
capture yes
linelimit       9999
interruptable   no
```

This job file records some or all of the following properties for the job (contents vary according to their relevance to the job):

jid	Job ID of the DJM job.
uid	User ID of the person running the DJM job.
sid	Session ID of the DJM job.
stated	What state the job is in (running, queued, halted, etc).
jobname	Name of the DJM job.
queue_name	Name of the queue the job is running in.
uname	Name of the user running the DJM job.
file	The executable file being run.
cdir	Current working directory.
stdout_name	The name of the file or device to which the job's standard output will be sent.
shell_name	Name of the shell in which the job is running.
pri	Priority of the job.
when_queued	Timestamp of when the job was submitted.
when_run	Timestamp of when the job started execution.
gid	Group ID.
groups	Groups to which this user belongs.
foreign_pid	UNIX process ID if the job is foreign.
active_pid	UNIX process ID of the job.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

accum_cputime Number of seconds the job has been running.

active_memory Amount of memory (in total Kbytes) the job has consumed.

active_when_started
The time when the most recent CM process within the job began execution.

max_memory Maximum amount of memory (in total Kbytes) that the job has used.

is_timeshared Whether the jobs is running on a timeshared partition.

actual_nproc Number of physical processors the job is running on.

actual_baseproc
The base processor address of the partition.

requested_cputime
How many seconds of CPU time the job requested.

requested_memory
How much memory (in total Kbytes) the job requested.

requested_nproc
Number of processors the job requested.

requested_baseproc
The address of the base processor the job requested.

dedicated Whether the job requested dedicated access.

rerun The maximum number of times this job may be rerun after an abnormal termination.

sp_disk Space being requested on the scalar processor (i.e., CP) disk.

sp_memory Memory being requested on the scalar processor (i.e., CP).

server_name FQDN (fully qualified domain name) name of the server the DJM job is running on.

Copyright © 1993 Thinking Machines Corporation

server_interface
: Number of the interface on the server the job is running on.

foreign
: Whether the job was submitted outside of DJM.

hastty
: Whether the job has an attached tty.

echo
: Whether keystrokes on an interactive job are echoed back to the user.

capture
: Whether output is to be sent to both the terminal and the file specified by the **-stdout** option to **jrun**.

linelimit
: Number of lines that can be sent to **stdout**.

exact_nproc
: Whether the job must run on exactly the number of processors it requested.

interruptable
: Whether the user has marked this job (via **-interruptable**) as being safe to kill before it completes. Jobs that are interruptible are allowed to begin execution even if they request more time than exists before a scheduled shutdown.

njobs_queued_ahead
: The number of jobs this user had queued ahead of this job.

Copyright © 1993 Thinking Machines Corporation

Appendix F

DJM Configuration File

The DJM system configuration file defines the machines that comprise the system, establishes pathnames, etc. You must edit the file to make it conform to your site. Lines beginning with a "#", tab , space, or newline are ignored.

This appendix explains each field in the DJM configuration file.

F.1 sitename Field

When DJM presents the output from a job, a banner is created that contains the name of the site. Specify the sitename here, 32 characters maximum.

```
sitename Thinking Machines Corp
```

F.2 machine Field

A single configuration file can define multiple machines. A "machine" is usually a CM, but may be a partition of a CM (as explained below). Each machine has a name (used to identify the machine to DJM daemons), a **`CM_name`** (used for CM2s and CM-200s to obtain information from the time-sharing daemon), a type (**`cm5`**, **`cm2`**, or **`cm200`**), memory per processing node in Kbytes, number of processing nodes, and the minimum number of processing nodes that may be accessed on the machine.

Copyright © 1993 Thinking Machines Corporation

Note that on a CM-5, the number of processing nodes is not the number of physical nodes, but rather the number of NI addresses available on the system. This allows one to define a partition anywhere in the CM address space. The number of nodes can easily be determined by raising 4 to the power of the CM network height. For example, a height 5 CM will have 4^5 (= 1024) network addresses.

```
#         name       CM_name    type   kbmem/pn   nproc    min_nproc
machine   exp-sparc  exp-sparc  cm5    16384      128      64
machine   exp-vu     exp-vu     cm5    32768      128      64
```

Two machines are defined here, one named **exp-sparc** and another named **exp-vu**. Both are CM-5s and each has 128 NI addresses. The minimum number of PNs that may be allocated on either is 64. Finally, **exp-sparc** has 16 Mbytes of memory per PN, while **exp-vu** has 32 Mbytes of memory per PN.

Note that **exp-sparc** and **exp-vu** actually are part of the same CM-5. One partition runs with VU chips, and the other without VU. However, DJM machines must be homogeneous with respect to VU or non-VU status. Hence, we configure two machines, even though they reside in the same physical CM-5 cabinet.

F.3 master Field

Each CM requires a master host, where the **djm_master** daemon runs. It need not be a front end or control processor. The name field must match a name supplied for the machine field. In addition, the keyword “master” must be followed by at least one space, so that the **djm_startup** script operates correctly.

```
#         name         Master Host
master    exp-sparc    mars.think.com
master    exp-vu       venus.think.com
```

Here, master hosts are defined for both the machines that were previously configured.

Copyright © 1993 Thinking Machines Corporation

F.4 server Field

The server field specifies the hosts that will be servers for each defined machine. Servers (which run **djm_fingerd** and **djm_execd**) must be CM front ends or partition managers. The machine name entry must match the name entry for the machine field, and the hostname specified must be the FQDN (fully qualified domain name) for the host. The master host and server host need the FQDN in order to create a TCP socket for communication between themselves. Finally, this field specifies the interfaces. For a CM-5, this is simply **0** (zero). The field is maintained for back-compatability with CM-2 and CM-200 DJM systems.

```
#      machine           FQDN       interfaces
server exp-sparc   mars.think.com         0
server exp-vu      venus.think.com         0
```

In this example the server for each machine is the same as the master host. The entry supplies the FQDN. Because the machines are CM-5s, the interfaces are specified as **0**.

F.5 client Field

The client field allows you to specify which hosts can submit jobs to a CM being managed by DJM.

Each client line contains the machine name, the FQDN of the client, and the hostname of the client returned from a call to the **hostname** command.

Since client commands (such as **job**, and its various subcommands) do not take a machine name as an argument, the client program must look at the configuration file to determine its CM machine. It does so by using the first machine for which it is listed as a client in the configuration file. In the example below, Venus is first listed as a client of machine **exp-vu**. Therefore, calls to **jstat** will default to machine **exp-vu**. If both client lines for **exp-sparc** preceded those for **exp-vu**, then **exp-sparc** would be the default CM machine for both Mars and Venus. Keep this in mind when editing the client fields.

Copyright © 1993 Thinking Machines Corporation

```
#       machine     FQDN                local_hostname
client  exp-sparc   mars.think.com      mars.think.com
client  exp-vu      venus.think.com     venus.think.com
client  exp-vu      mars.think.com      mars.think.com
client  exp-sparc   venus.think.com     venus.think.com
```

F.6 Directories and Files Fields

The location of a number of directories and files are defined in the configuration file. Edit these to meet the needs of your site.

tracelog: When DJM daemons panic, they dump a trace of their execution to the filename listed in the **`tracelog`** field. The **`%s`** has the name of the daemon substituted when the dump occurs.

```
tracelog /tmp/%s.trace
```

spool_dir: The location of the spool directory is indicated by the **`spool_dir`** field. The various queue configuration files are saved in this directory. There must be a spool directory on each server.

```
spool_dir     /usr/spool/djm
```

cookie_file: To provide for secure TCP messages, each TCP message has an authentication key. The daemons examine this key and reject messages that do not have the correct key. The authentication key is made up of two parts. One is stored on-line in a file (which, obviously, should have root-only read/write permissions). The **`cookie_file`** field indicates where this file is located.

Authentication is discussed more fully in Sections 2.1.6.8 and 2.2.1.

```
cookie_file /etc/cm/configuration/djm.cookie
```

default_path: Each job has a default path when started. Specify the path in the **`default_path`** field.

```
default_path /usr/ucb:/bin:/usr/bin:/usr/lbin:
```

default_umask: Likewise, each job has an initial **`umask`**. It is specified in the **`default_umask`** field.

```
default_umask 027
```


Copyright © 1993 Thinking Machines Corporation

F.7 Utility Programs Fields

DJM makes use of a number of utility programs to access NFS-mounted files, verify projects, etc. The location of these utilities is defined in the configuration file.

djm_shepherd: The location of the **djm_shepherd** process (defined in Section 2.1.1) is specified in the **djm_shepherd** field.

```
djm_shepherd /usr/etc/djm_shepherd
```

cknfs: DJM makes use of a utility to ping NFS servers before attempting to change directories. If the NFS server is dead, DJM does not try to change to that directory. This avoids having DJM hang because of a dead NFS server. The utility is available on numerous Sun source archives, title "V01i033". The source and binary are also included in the DJM release for your convenience. Specify the absolute path to the utility in the **cknfs** field:

```
cknfs /usr/local/bin/cknfs
```

fixpath: If your site uses the automounter, it is possible that pathnames returned from an automounted directory will contain kruft that needs to be removed before using. A utility script to do this must be written to meet the needs of your site. The **fixpath** field in the configuration file specifies the location of this script:

```
fixpath /usr/local/etc/fixpath
```

A sample script might be something like this:

```
#! /bin/sh

# Strip off automounter garbage from the start of pathnames.
# This garbage inhibits system calls such as chdir() from
# actually changing to the correct directory. IE, take
# /tmp_mnt/proj/foo/bar and return /proj/foo/bar, or
# take /proj/foo/bar and return /proj/foo/bar

sed 's/\/tmp_mnt//' << EoF
$1
EoF
```

In this case the automounted directories have a prefix of "**/tmp_mnt**" that must be stripped for **chdir()** to operate successfully.

Copyright © 1993 Thinking Machines Corporation

verifyproj: An optional field in the configuration file is for a script that verifies the project name of a job.

```
verifyproj    /usr/local/etc/verifyproj
```

gid fields: In order to correctly associate CM processes with DJM jobs, DJM assigns a group ID to the job. Consequently, there must be a range of group IDs available to DJM to use. The **`first_gid`** and **`last_gid`** fields of the configuration file specify the range of group IDs to use. It is imperative that there be no valid group IDs already in use in the range specified. The **`/etc/groups`** file should be consulted to be sure of this.

```
first_gid   31000
last_gid    32000
```

sendmail: DJM will send mail to users regarding their jobs. To do so, DJM must know where to find the appropriate mail program. The **`sendmail`** field identifies the location.

```
sendmail  /usr/lib/sendmail
```

contact: In some problem situations, DJM's message includes an electronic mail address to which users can send questions. Use the **`contact`** field to specify this address for your site:

```
contact djm-problems@site.com
```


Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

Appendix G

NQS Compatibility

DJM is compatible with the Network Queueing System (NQS) in terms of the user's view of job submission. Indeed, system administrators at sites that currently use NQS can configure the system in such a way that all user-level NQS commands (**`qsub`**, **`qlimit`**, **`qdel`**, and **`qstat`**) continue to work as they did previously.

This appendix summarizes the points of similarity, difference, and compatibility between the user's view of job submission with DJM and NQS.

G.1 Queues

Unlike NQS, DJM relies on a small number of queues to manage the job flow. DJM has no need for pipe queues, as it performs load balancing automatically.

In addition to the expected queues, DJM maintains a foreign queue. This queue handles jobs that are not submitted via DJM commands. (From a somewhat cynical standpoint, one could say that this queue traps jobs whose users are trying to avoid DJM.) The administrator can impose limitations on the foreign queue to make it unattractive, and thus to force the use of DJM.

Copyright © 1993 Thinking Machines Corporation

G.2 qsub and jsub Options

The following options to **jsub** are supported for compatibility with the NQS command **qsub**:

-e *name* — Sets the standard error file to *name*. Same as **-stderr**.

-o *name* — Sets the standard output file to *name*. Same as **-stdout**.

-eo — Combines standard error and standard output files. This is the default in DJM.

-ke -ko -re -ro — Ignored. DJM makes no distinction between local and remote output.

-lT *n* **-lt** *n* — Sets the per-job memory limit to *n*. Per-process limits are not supported by DJM.

-lF *n* **-lf** *n* — Sets the per-job front-end disk space limit to *n*. Per-process limits are not supported by DJM.

-mb -me -mu *user* — Sends mail at the beginning and end of the job, to user *user*. Same as **-mail_begin**, **-mail_end**, and **-mail_user**.

-q *name* — Submits the job to queue *name*. Same as **-queue**.

-r *name* — Sets the job's name to *name*. Same as **-jobname**.

-s *name* — Specifies the shell to interpret the job's shell script. Same as **-shell**.

-x — Exports all environment variables to the job. Same as **-export**.

-z — Submits the requests silently. Same as **-silent**.

Version 2.0, November 1993
Copyright © 1993 Thinking Machines Corporation

G.3 Submission Requirements

DJM requires you to estimate a minimum of three resouces to manage job submission successfully. These resources are: number of processors required, CPU time required, and memory required. NQS does not place this requirement on users.

G.4 Shells

DJM's shell strategy is equivalent to NQS's concept of free and login shell strategy. DJM does not contain a counterpart of NQS's fixed shell strategy.

Copyright © 1993 Thinking Machines Corporation