Maximising Job Throughput Under NQS on the CRAY T3D

Michael W. Brown, Ediburgh Parallel Computing Centre, The University of Edinburg, King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, United Kingdom

Abstract

The University of Edinburgh operates a 320-processor Cray T3D on behalf of the UK academic and research community. The system is used principally as a batch resource for large-scale production runs, but also supports development access for emerging projects as well as satisfying the needs of users with less than full "grand challenge" applications. The requirements for a batch system differ between MPP and PVP systems, and this paper describes how the NQS configuration has been optimised under experience and the evolving workload to maximise throughput until around 96% of possible batch cycles have been delivered to a large and disparate user population.

Site Overview

On behalf of the UK Engineering and Physical Sciences Research Council (EPSRC), the managing agent for academic supercomputing in the United Kingdom, the Edinburgh Parallel Computing Centre (EPCC) at the University of Edinburgh has run a peer-reviewed national service for Grand Challenge science in a Cray T3D system to the British academic and research community since July 1994. A report **[1]** on the first year of the scientific programme being undertaken on the machine will be published in October.

The original 256 processors and the Y-MP4E front-end system were installed in April 1994, and the configuration received its first substantial upgrade in December 1994 when an additional 64 processors were installed in the T3D.

The operation of the service has already been described **[2]** but there have since been a number of substantial changes to the service planned recently, and these were publically announced at the EPCC's Annual Seminar in Edinburgh on September 21st.

The principal announcements included the upgrading of the T3D parallel resource by a further 64 processors (bringing the total to 384 processors) and the augmentation of the front-end resources (currently a heavily-used Y-MP4E system) with a 10 processor J90 series machine with 1 Gb of main memory. The filesystems on the Y-MP and J90 will be shared using DFS across a HIPPI link. The total disc resource will be more than doubled to give a total of 500 Gbytes of on-line disc capacity available to MPP users and it is planned to increase the capacity of the IBM 3494 automatic tape library to in excess of 20 Tbytes by year end with the replacement of the existing 3490E tape technology with 3590 (NTP).

Hardware configuration

sn6001:
T3D/MCN384-8 (384 PE, 8 MW/PE)
2 x IOG's

sn1904:
Y-MP4E/364
32 MW SSD
2 x IOC's
36 x DA301 (197 GB) on 8 x DCA-3
8 x DD60 (15 GB) on 8 x DCA-2
1 x HIPPI on 1 x HCA-3/HCA-4
2 x FDDI on 2 x FCA-1

sn9192:
J916/10-1024
5 x IOS's
8 x DDS-30 (288 GB) on 8 x SI-3
1 x IBM 3494 ATL system on 2 x SI-3X
1 x HIPPI
2 x FDDI
1 x ATM

User Population

The user population of the T3D is comparatively large and disparate. Time is allocated by a peer-reviewed allocation panel to consortia who are required to bid for resources. The allocation panel meets every six months, and allocates time to each consortium on the basis of the strength of their case, the recommendations of those who have independently reviewed their application, and on their track-record on the machine of work on the machine to date.

Some of the consortia are large, they may contain some dozens of researchers from institutions from all over the country who are working together in a particular area of science, and they may have a science programme that requires very substantial amounts of computing resources.

Other consortia are much more modest in total size, and in their aspirations for time on the machine, and already some consortia who started off with modest requirements from the T3D have been able to expand their applications and, after effective use of early-use time, have become able to efficiently use larger proportions of the resource.

At the present time there are about seven consortia who are considered to be in the large category, and they typically are awarded in excess of 5% of the total amount of cycles for a six-monthly period (the two very largest receiving around 18%). There are about 20 consortia who are in the more modest category, and are typically awarded between 1% and 5% of the total number of cycles, and there are a further 10-15 consortia who may be awarded much more modest amounts of processing time for "pump-priming" purposes, typically to enable them to progress their science sufficiently so that they can make a full application for time subsequently.

The overall utilisation of the T3D has grown to substantial levels, with a utilisation in excess of 70% on seven out of the first eight months in 1995 with a peak utilisation of 89.37% in May. 295 individual users from in excess of 30 active user groups have accessed the machine, with 1.35 million processor hours being used until the end of August. A summary of the utilisation is given in table 1.

Month	PE hours	Util.	Users	Service.
Jan	103512	50.64%	122	99.78%
Feb	157546	75.12%	135	98.79%
Mar	201460	85.25%	136	93.92%
Apr	184404	87.78%	132	99.46%
May	204975	89.37%	133	97.90%
Jun	171644	73.08%	141	99.31%
Jul	154197	83.88%	145	90.87%
Aug	180595	84.42%	156	93.19%

Table 1

Up until very recently, the production-time usage has been dominated by around three user groups, although now resources are being used more equitably and while in August no user group used more than about 15% of the total cycles, six were able to use in excess of 5% each and 29 groups (156 users) in total used the MPP to some degree.

In addition, the University of Edinburgh has 20% of the cycles on the present (320-processor) system allocated for its own use to reflect the investment

made by the University when the system was upgraded in December 1994.

User Job Requirements

The T3D was purchased by the British government to enable researchers in the UK to undertake world-class science in a number of key "grand challenge" areas (such as the modelling of climatic change, simulations of materials, particle physics, and computational fluid dynamics). It is clear that the bulk of the processing resource must be made available to those tackling such problems, and for whom there is no other suitable computing resource available. That being said, it was rapidly made clear that the concept of an 'exclusive club' of user groups was out of line with research council policy to enable access to such a resource to smaller, less demanding, but still suitable groups of users, and that sufficient resources must be made available to new and evolving scientific programmes as well as the the existing established ones.

The ability to effectively partition the resource to enable smaller applications to run concurrently with larger ones, as well as to provide a code-development resource was essential and some of the short-comings in the original 256 processor configuration were thus addressed when the system was subsequently upgraded to one of 320 processors. The continued expansion of the science programme has created a scarcity of resources again, so the further upgrade to 384 processors is especially welcome.

Job Size Profile

On the T3D system, jobs require processors in powers of 2, thus on the Edinburgh configuration the smallest job will use 2 processors and the largest 256.

Users are encouraged to use the largest number of processors that is appropriate to their application, and generally 'production' jobs requiring modest numbers of processors are not welcomed as their impact on the system may cause fragmentation (see below) and there may be more appropriate platforms upon which such work may be run. Jobs of modest size and duration are obviously required for code development purposes, and the operational regime in place enables this.

The NQS configuration (see table 3) is optimised for dealing with jobs requiring 64 or more processors, and interactive access is enabled only for jobs requiring 32 processors or less, except for system test jobs.

The following table gives the job size profile for the month of August, it should be noted that each 'job' referred to is actually every call of 'mppexec', and that many batch jobs consist of multiple calls of 'mppexec' (both sequentially and concurrently).

PE	Inter	Batch	Users	PE Hours	%ge
2	4453	36	78	193	0.11%
4	2518	11	76	160	0.09%
8	2384	14	61	655	0.36%
16	2204	42	77	1161	0.64%
32	1783	81	64	5987	3.32%
64	2	1455	68	45984	25.46%
128	0	760	42	25985	14.39%
256	6	855	37	100471	55.63%

Table 2

As can be seen, the overwhelming amount of time on the T3D is used by applications running in batch (96.42%), and the configuration is optimised very much towards supporting batch use. The requirement to make an interactive service available does impact to some degree on the total throughput of work on the machine, but this is unavoidable if development access to the system is to be retained.

The overwhelming proportion of MPP time used on the system is for jobs requiring at least 64 processors. This justifies the choice of an MPP platform, and any sort of production work that requires less than 64 processors is actively discouraged from using the resource as the system was procured as a vehicle upon which to undertake the very largest scientific simulations and was not just to be considered as an overgrown workstation farm. The implementation of this policy means that less than 5% of the cycles are used by applica-

tions requiring less than 64 processors. Continuing reduction in throughput due to fragmentation (see below) caused by long-running batch jobs requiring small numbers of processors has meant that a policy of forcing such jobs into queues that may prevent, or substantially delay, their execution is being considered.

The numbers of users that are able to exploit such a large resource is large and growing, although the running of production applications requiring 256 processors is currently limited to originating from around 7 of the largest user groups.

Machine Configuration

The 320 processor machine is configured into two pools, one containing 256 processors and the other containing the remaining 64 processors.

The 256 processor pool is generally used for batch work, with the 64 processor pool being used for interactive access during the day.

There are three configuration periods,

- DAY (1000 - 1800 Monday-Friday)
- NIGHT (1800 - 1000 Monday - Thursday)
- WEEKEND (1800 Friday - 1000 Monday)

with the following details:

DAY configuration

The 64 processor pool is configured solely for interactive use. No user may request more than 32 processors, and the MPP residency time is limited to 10 minutes.

The 256 processor pool is configured for mixed batch/interactive access with priority given to batch. Job sizes of 128 and 64 processors are supported within the NQS queuing structure, with a maximum job-residency time of 1 hour. Typically, 1 x 128 and 2 x 64 processor jobs are run concurrently, and special 'short' queues (for jobs requiring less than 20 minutes MPP residency time) are given priority. If there are unused resources within the pool (typically because a submitted job required only 32 processors), interactive users may access those processors remaining within the pool. This was done to increase the total development resources during peak times when demand may be highest for short-turnaround interactive jobs, but there is a penalty to be paid in the increased chance of pool fragmentation leading to unnecessary idle cycles.

Any batch applications still running at the close of the DAY configuration are aborted, but interactive jobs will flow through the configuration change period at 1800 hours.

NIGHT configuration

The 64 processor pool is configured solely for interactive use until 2000 hours, when it becomes configured solely for batch use. At 0800 hours, sole interactive use is enabled again, these two periods being made available at the request of the users so as to give development access in the early part of the working day, and into the evening which suits many research workers.

The 256 processor pool is configured solely for batch use (principally for 256 processor jobs), although smaller jobs will be run when 256 processor work is exhausted. On Monday nights the queue order is reversed to enable any backlog of smaller (principally 128 processor) jobs that may have built up to be dealt with.

The maximum job-residency time for both pools is 6 hours, but a special 'short' 256 processor queue is given priority for jobs with a maximum MPP residency time of 20 minutes up until 2000 hours. This was configured, at user request, as potential users of the 256 processor resource had no other opportunity to make scaling runs at the full-size of their application, and the job backlog on the 256 processor queue has been in excess of one week.

WEEKEND configuration

The configuration run at weekends is similar to that of the NIGHT configuration, except that no jobs from the 'short' 256 processor queue are run, the maximum MPP job-residency time is increased to 12 hours, and interactive access only is granted to the 64 processor pool between 1000 and 1800 on Saturday and Sundays so as to allow some degree of development access over the weekend. The WEEKEND configuration is considered to be a special case of the NIGHT configuration, and jobs are submitted to the NIGHT pipe queue which routes those with the necessary time requirements to the queues that only run at weekends.

NQS Configuration

The NQS configuration on the Y-MP front-end is directed almost entirely towards supporting job execution on the T3D and the only other type of work that is currently supported is compilation work associated with use of the MPP. The provision of the J90 system shall have a significant impact on the amount of non-MPP work that can be accommodated within the entire configuration.

Currently, there are two pipe queues set up, called 'DAY' and 'NIGHT'. These pipe queues feed a number of MPP batch queues that differ only in the maximum number of processors that the job may use, and the total job-residency time. The 'WEEKEND" configuration is seen as a special case of the 'NIGHT' configuration, and a separate pipe queue was not set up.

Users submit their jobs either to the 'DAY' or 'NIGHT' pipe queues, and NQS will route the job to the appropriate batch queue according to the `#QSUB -l mpp_p` and `#QSUB -l mpp_t` settings. If no such settings are made, the job will fall through to the default queues COMPILE and COMPILE_NIGHT which have a maximum Y-MP CPU allocation of 1 hour and 6 hours respectively, and which are used only for MPP related tasks, typically compilation.

Y-MP specific limits (such as CPU time or memory allocation) are not generally applied to the MPP batch queues. The limits are set up as described in the section that describes the operational configurations, and are summarised in table 3.

Queue	PE's	Time	Configuration
MPP_S64	64	00:20	DAY
MPP_S128	128	00:20	DAY
MPP_S256	256	00:20	NIGHT
MPP_64	64	01:00	DAY
MPP_128	128	01:00	DAY
MPP_256	256	06:00	NIGHT/WEEKEND
N_MPP_64	64	06:00	NIGHT/WEEKEND
N_MPP_128	128	06:00	NIGHT/WEEKEND
W_MPP_64	64	12:00	WEEKEND
W_MPP_128	128	12:00	WEEKEND
W_MPP_256	256	12:00	WEEKEND

Table 3

If an MPP job is specified with a total residency time of in excess of the current maximum (12 hours), it will be routed to a special holding queue that is never allowed to run. This allows such jobs to be identified and the users informed, and also enables very long jobs to be manually scheduled on request.

All jobs are aborted at the conclusion of each configuration period, the bulk of the users accessing the system are able to make their own arrangements with respect to checkpoint/restart, and it was viewed as essential that jobs were not allowed to overflow from one configuration period to the next and cause subsequent disruption.

Future changes

It is clear that the commissioning of the J90 system, and the additional 64 processors, within the overall configuration will require substantial changes to the NQS configuration. Currently, it is planned to have pipe queues configured on each host, with one on the J90 feeding MPP jobs submitted there to the Y-MP, and one on the Y-MP feeding compilation, or post-processing jobs to the J90. It is anticipated that the J90 system will absorb the bulk of the work that is running on the Y-MP that is not directly related to the execution of a job on the T3D, and the significant increase in front-end CPU capacity will allow effective post-processing of MPP-generated

data to be undertaken within the same machine configuration. Until now the bulk of such work has had to be routed to remote facilities putting an additional load on those facilities as well as straining the network to cope with the potentially very large data-transfer requirements.

The 384 processor system will be divided into two pools, one containing 256 processors and one containing 128 processors. As the largest job that can be accommodated on the 384 processor T3D will remain at 256 processors it is not envisaged that there will be substantial changes to the batch queues that feed the T3D, except that the simultaneous job limits for 64 and 128 processor jobs, and global PE limits, will increase where appropriate.

NQS Limitations

There are a number of limitations within NQS as supplied for the T3D system which can have a significant effect on the total throughput of MPP work.

Fragmentation

Job scheduling for MPP systems is an inherently difficult problem. In a traditional computing environment batching software only has to worry about total usage of resources. For example, the total number of CPU seconds used and the maximum amount of memory used at any one time. This is sufficient because the operating system is responsible for dividing the CPU time and physical memory between all the active processes.

On an MPP system the primary resource to be allocated is the PE. PEs are not shared between running programs and they must be allocated in contiguous blocks. It is therefore insufficient to just consider the total number of PEs used by batch jobs. Even when there are sufficient free processors a program may not be able to start because the free processors are in different areas of the machine. It follows, therefore that a job may have a greater impact on the total throughput of the machine than the resources it itself consumes, and this has been regularly observed on the Edinburgh T3D.

This problem can cause significant wastage of resources on an MPP system if very small jobs are allowed to run in concert with larger jobs. On the T3D jobs are fitted in to the torus according to their 3-dimensional shape which is fixed for each size of job. On the Edinburgh T3D, 64 processor jobs have an XYZ shape of 16 x 4 x 1, thus all jobs requiring 64 processors or less are "flat" in the Z-dimension. Small jobs starting in the area occupied previously by a 64 processor job can effectively prevent that starting of larger jobs. On the Edinburgh system, during the DAY configuration, normally 1 x 128 and 2 x 64 processor jobs are configured to run. If a 128 processor job completes, and there is not another to fill its place, a further 64 processor job may be able to start in its place. Even when subsequently there are 128 processors available, they may be in 'slices' of the machine such that the default shape (16 x 4 x 2) cannot be placed. What makes this so serious a problem, is that NQS is unaware of the current allocation of resources, only the total number of processors that are currently allocated. Thus, NQS will happily start a 128 processor job (because there are 128 processors free), but the job could then block (and block any other job from starting), because it could not place the shape upon the torus.

It seems quite incredible that the linkage between NQS and the software in UNICOS-MAX that is responsible for the allocation of resources and the placement of allocated resources upon the torus totally overlooks this problem.

To maintain a high machine utilisation the batch system has to operate in such a way as to reduce fragmentation to a minimum. It is particularly important to reduce fragmentation because fragmentation reduces the overall throughput of the machine and restricts the size of job that can be run at any one time. If the machine is allowed to fragment, large applications that require large numbers of processors will be unfairly penalised. Regrettably, the lack of effective resource-availability checks between NQS and the MPP system, means that the operating system is unable to help to reduce

processor fragmentation.

Historically memory fragmentation used to be a significant problem on conventional systems but this ceased to be the case when virtual memory allowed memory to be reallocated while a program is running. Even on real-memory machines like the Y-MP, the operating system can remove memory fragmentation by swapping processes out to disc and then restarting them at a different location in memory or by relocating a running process in memory. The T3D currently does not provide any mechanism to do either of these operations.

Cray have very recently added a mechanism to roll a MPP program out to disc. This feature, released in UNICOS-MAX 1.2.0.5, will be slow and require large amounts of disc space to hold the program images, and it is difficult to see how this feature can be of use except to enable the system administrators to perform some manual de-fragmentation by identifying small jobs that are effectively blocking the placement of larger jobs and rolling them out, and subsequently back to a location that causes less interference.

In principle it should be possible to extend UNICOS-MAX to also include the facility to relocate a running process to a different base-PE. This is not significantly different from the roll-out facility already being implemented though it should move data directly between PE memory using the internal communication network rather than creating an image file. If the operating system removed fragmentation by automatically relocating processors, the scheduling problem would become significantly easier.

Premature job termination

Once NQS is running, it possesses no knowledge of when it might be closed down, thus it is quite possible for a job to be selected to start only a few minutes before a planned shutdown of NQS for a configuration change (say from the NIGHT to the DAY configuration).

If NQS possessed such knowledge, it could make more intelligent job selection, thus if there was only one hour left before a planned shutdown the queues could be scanned for a job of the appropriate length that could be run until the configuration change. Batching software should not normally be permitted to start a job if it cannot be completed within the time left for the configuration to run, unless the user is happy to take the risk of an aborted job (as many users perform their own checkpointing, the loss of time may not be critical, other users can only write out their output data set at the end of a run, which may be of some hours duration).

One solution is to run with an increased number of queues (with different maximum job-residency times), and start to shut the queues down as the time for the closedown period approaches. The main problem with this is that it can lead to idle time and shorter jobs may get locked out from running at all as additional longer running jobs get submitted in the period say between one NIGHT configuration and a subsequent one (longer jobs would be given priority at the start of each session).

Alternatively, if NQS is allowed to start up jobs right until a closedown period, it reduces the amount of resources left idle, but at the expense of terminating user jobs prematurely. Users get justifiably irritated when a job they have been waiting to run for a week climbs to the head of the queue only to be aborted immediately after it is started due to a planned configuration change.

Job selection criteria

NQS uses a very simple first-in-first-out scheduling policy. This causes particular problems in that it can allow a single user, or group of users, to unfairly dominate the queues, and thereby the work that is run on the machine. NQS does not allow limits to be applied on the number of jobs that may be submitted by a particular user, or group of users, and it is quite possible for a user to submit a whole series of jobs just before departing on vacation that would, if allowed to run in sequence, dominate the usage of the system. One user at Edinburgh recently submitted 400 jobs at once.

It would be far preferable if the systems administrator was able to apply limits (in the UDB for

instance) that could put a limit on the total number of jobs that a user could submit, or on the total amount of MPP time (PE hours typically) that the user had queued to run.

The Edinburgh T3D supports a wide variety of user groups with different resource requirements and allocations. It is relatively easy to ensure that each group has a fair allocation of resources. MPP time is allocated in PE hours to groups, and the Principal Investigator of each group assigns time to individuals or sub-groups. The actual amount of time is allocated to ACIDs, and usage is clocked against the total amount left in the ACID. When the amount remaining in the ACID reached zero, any attempts to use the MPP with that ACID as the current one is denied. It is not so easy to ensure that each group has fair access to use these resources.

On a conventional machine, it would be sufficient to ensure that each user or user group has a fair share of the available CPU resources. It is more complicated on a MPP system. The availability of the machine will be different depending on the size of job that is to be run. The machine has to support a mix of different job sizes and this balance has to be carefully maintained. At one extreme, relatively small or short batch jobs may be denied access to the machine if it spends too much time running the larger jobs. At the other extreme, if priority is given to smaller jobs the machine may become fragmented preventing the larger jobs from running. In this case, processors are left idle and the overall throughput of the machine is reduced.

The maximum job-residency time chosen for the DAY, NIGHT and WEEKEND configurations, plus the availability of the 'short' queues for jobs of each main production size (64, 128 and 256 processors), has to some extent been a compromise between enabling major user groups to gain access to serious amounts of production time, whilst still allowing access for smaller or emerging projects.

With the configurations currently in use, large jobs (64 processors or greater) can be managed for large amounts of time, large jobs for small amounts of time (less than one hour), small jobs for small amounts of time, but not for small jobs requiring large amounts of time. It is the latter case that causes fragmentation, and EPCC believes that such work is not in any case inappropriate for a running on a large MPP system and such use ought to be discouraged.

Different numbers of processors are applicable to different calculations. On the Edinburgh T3D not only does it have to be ensured that the right mix of 64, 128 and 256 processor time is provided but that this time is divided appropriately between the different groups competing for that class of time.

Over an entire time allocation period (currently six months, with two month sub-allocation periods between which users may request time to be moved forward, but not back), the resources consumed by user groups will generally be in line with the allocations (unless a group fails to make use of their allocated time), the major problem is ensuring that no single group can distort the usage on the machine on a short time frame (like a few days, or a week at a time) by filling the queues. Thus, fairness can only be ensured in a global sense by the time allocation procedure.

Jobs containing multiple runs

MPP jobs under NQS have two principal variables that are used to determine when and where these jobs are able to run. These variables are the number of processors (`mpp_p`) and the job residency time (`mpp_t`). Unlike a PVP system which shares CPU resource, the "CPU" time for an MPP job is not significant, instead it is the elapsed wallclock time that the job will require that is important. The actual CPU time is, of course, the elapsed time multiplied by the number of processors, but this is not used in scheduling as it is plain that 256 processors for two hours is not the same as two processors for 256 hours. Although the global resources consumed in each case would be the same, such jobs would need to be treated in very different fashions.

It would be expected thus, that jobs which actually consume identical resources during their period of execution would be treated in the same manner by the batch system, even if those same resources were used in slightly different ways within the jobs.

Regrettably, this is not necessarily so.

Many of the jobs run under NQS on the Edinburgh system, actually consist of multiple calls of 'mppexec', running either sequentially, concurrently or a mixture of both.

One major user of the T3D runs, within the same NQS job, a series of 256 processor jobs, 2 x 128 processor jobs and 4 x 64 processor jobs as it is more efficient to divide up different phases of the calculation being undertaken in this manner. This is an entirely appropriate use of the system, except that there is an inconsistency within NQS that divides the total MPP residency time up between concurrent calls of 'mppexec'.

It would be expected that the running of the following two job scripts would result in the jobs enjoying the same total MPP residency time (as identical resources are requested for use in each case).

```
#QSUB -l mpp_p=256
#QSUB -q night
#QSUB -l mpp_t=6:00:00

mppexec -npes 256 job.256
```

and

```
#QSUB -l mpp_p=256
#QSUB -q night
#QSUB -l mpp_t=6:00:00

mppexec -npes 128 job1.128 &
mppexec -npes 128 job2.128 &
wait
```

In fact, while the first job would run for six hours (as expected), the second would only run for three, and although an SPR has been raised with respect to this idiosyncrasy, it has not yet been accepted by Cray as a problem.

Configuration Optimisation

As the workload on the system has increased, then the problems outlined above (fragmentation, unfair job-selection, and premature job termination) have had an increasing impact on the throughput of the system and on the ability of the users to undertake their work.

Accordingly, changes have been made to the configuration and these are discussed below.

Configuration evolution

When the T3D was first installed, a very simple NQS configuration was implemented that was based on projected user job profiles. The job profile has been monitored since the start of service, and the NQS configuration adjusted to suit changing user requirements. Inevitably, this has led to a more complex NQS configuration, but the basic interface to the users has remained constant and many of the subsequent adjustments have been quite transparent to the user population. Many of these changes have not actually resulted in improved job turnround as the user load has increased, but the overall efficiency of the use of the resources has improved to the currently very high levels.

The following requirements were specified before the start of the service, and this have largely remained appropriate:

1. The machine was procured to enable researchers to use the largest resources available, thus for the bulk of the time the resource must be available for running production jobs that may require the largest possible proportion of the whole machine.
2. Some user groups may not be able to make most efficient use of 256 processors, but production runs requiring smaller numbers of processors (64 and 128 processors) must be supported.
3. Access to significant numbers of processors, for modest amounts of time, is required for code development purposes particularly for the smaller, or emerging user groups.
4. Interactive access to a proportion of the machine is required during the currency of the

normal working day.

The original 256 processor system was configured by day with three pools; one containing 128 processors was used exclusively for batch work, one containing 64 processors for shared batch and interactive and one containing the remaining 64 processors and used exclusively for interactive work. This configuration was quickly found to be non-optimal, with the 128 processor pool often idle due to an intermittent batch load early in the life of the machine, while the shared batch/interactive pool could become badly fragmented by small interactive jobs. This latter problem was exacerbated by the poor initial job placement algorithm that utilised a simplistic "first fit" strategy, rather than the "best fit" strategy adopted from UNICOS-MAX 1.1.0.4. and the major problems to be caused by fragmentation were not imagined at the start of the service.

After a short period the 3-pool configuration was replaced by one containing a single 256 processor pool, and access was regulated by applying a 192 processor global limit for NQS jobs. This enabled better utilisation to be achieved, with the interactive portion being able to expand dynamically into the area that was previously reserved for exclusive batch use. With minor adjustments, access to the 256 processor pool by day has remained following this broad pattern.

The original NIGHT configuration utilised only a single pool, and was enabled solely for batch use with priority given to 256 processor jobs.

Subsequent to the upgrade to 320 processors in December 1994, a number of changes were made to the configurations to reflect the increased size of the resource, and initially a 320 processor pool was configured for the DAY session, with a 256 and a 64 processor pool for the NIGHT configuration. The 320 processor pool was found to get badly fragmented by ill-placed small interactive jobs, so the decision was made to run with two pools, and thereby in effect separate the batch from the interactive work which reduced, although did not totally remove, the fragmentation problem with respect to the batch work. This change also had the added benefit that even if a batch job started in the 256 processor pool, and was then blocked due to fragmentation, the blocking (which is only global to a pool) would not prevent interactive jobs from starting within the 64 processor pool.

Adjustments made to the configurations since the start of the service have also included:

1. Enforcement of the '-nosleep' option for interactive calls of 'mppexec'.

   By default, if an interactive job could not be placed, it would block this, and all subsequent interactive requests, until the blocking request could be satisfied. This meant that an attempt to run a 32 processor application might fail as the job could not be placed (either due to lack of resources or fragmentation), and this would block subsequent jobs that could be placed. On a busy system, a long queue of interactive jobs in WAIT state would build up causing user frustration and wastage of resources. If an interactive job cannot be started, the request is now immediately terminated.

2. Matching job queues to the most common sizes of production job.

   This process has been helped greatly by the "powers-of-2" rule with respect to job sizes;

3. Introduction of the 'short' queues for jobs requiring less than 20 minutes MPP residency times.

   This has enabled brief proving runs to be undertaken in advance of full-scale production, and has enabled effective development work to be done without impacting upon the large-scale production service;

4. Introduction of the 6 and 12 hour limits for production jobs overnight and at weekends.

   This was done to maximise fairness (otherwise the system could be used in exclusive mode by a user group for a complete night or weekend session);

5. Altering the DAY configuration to allow for 1 x 128 and 2 x 64 processor jobs to run concurrently;

6. Changing the job queue priorities so that, for instance, 128 processor jobs are given priority one night per week to avoid such jobs forever being pre-empted by 256 processor work.

Improving job selection

As it is not possible to apply, using UDB limits for instance, any form of limit to the number of jobs that may be submitted by a user, it was necessary to come to some solution of the problem whereby a small subset of users could gravely distort the usage of the system by flooding the system with work.

Initially, the job queues were manually re-processed, so that the user who submitted multiple jobs would find that the bulk of them had been placed further down the queue, the problems with this approach became clear as the total number of users and jobs increased. It was believed that some user groups were avoiding penalty by dividing their production work up between multiple user ID's, and the perceived unfairness was still present.

It was decided to automatically reorder the NQS queues, and to do the reordering based on consortium membership, rather than by username. This was a fairer way to do it, and was fairer in any case on the users who may be a member of more than one user group.

As all MPP resource allocation and accounting on the Edinburgh machine is done according to ACID, the reordering was done according to the ACID value specified in the NQS job.

Each batch queue is resolved into a list of the following format:

```
18796   a2
18798   a2
18799   c1
19377   c1
19378   e5
19384   f1
19392   h1
```

where the first column is the NQS job id, and the second is the identity of the user group according to the ACID value. The entries for user groups who divide their MPP allocation into multiple ACIDs (such as c1-prod and c1-dev) will be resolved down to the basic unique group id (in this case c1). The list is then sorted so that multiple jobs for the same group are cascaded down the list, but the execution order of jobs belonging to the same group is preserved as typically one job will produce the output for its successor.

The result of reordering in this case is:

```
18796   a2
18799   c1
19378   e5
19384   f1
19392   h1
18798   a2
19377   c1
```

Currently, the reordering is made just before the start of each DAY and NIGHT configuration period, but it can be initiated manually at any time to reflect the changing rate of job submissions during the day for instance.

Avoiding premature job termination

A major source of user irritation is the forced premature termination at the end of each DAY/NIGHT configuration period caused by the inability of NQS to select a job that will fit in with the remaining time within a configuration period.

EPCC has provided a simple facility called 'timecheck' which can be called by a user at the start of a job. It derives the current time, and knowing the requested job duration reads an entry in a file that contains the scheduled time of the next closure and will return a result code that the job can interpret and cause it to re-submit and terminate.

The problems with this are twofold:

1. As the job has already started before it makes the 'timecheck' interrogation, the job has lost its place at the head of the queue to which it might have taken up to a week to reach.

2. A job cannot pre-determine which pool it will run in, and the 64 and 256 processor pools currently have different termination times for batch access (the 64 processor pool is deconfigured at 0800 and the 256 processor pool at 1000 each weekday for instance). Thus, a 64 processor pool may get a positive response from 'timecheck', but the job may still abort early.

Nevertheless, the facility has proved useful, especially for users running 256 processor applications, as it can avoid the total wastage of a significant amount of allocated time (as some applications must complete in good order to be of any use) with the penalty of losing the place in the job queue.

It is clear that NQS as it stands is unsatisfactory in this area, and EPCC is considering providing a user exit to interface with NQS that will be called at the time of job selection and which will be able to make an intelligent decision on the desirability, or otherwise, of letting NQS start a job close to the configuration closedown time. The principal advantage of this approach is that the test will be made in advance of the job being started, thus the job does not lose its place in the queue, and the facility may be extended to include other tests that may be used to increase the overall fairness of the job selection process.

DJM - Distributed Job Manager

DJM, the Distributed Job Manager **[3]**, is a job scheduling system developed by the Minnesota Supercomputer Center specifically as a scheduling system for MPP platforms.

It was developed originally for Thinking Machines Corporation CM-2 and CM-5 systems in view of the unsatisfactory scheduling undertaken by NQS, and was subsequently modified to support the Cray T3D.

DJM is fairly sophisticated, and the implementation for the T3D has three principal advantages over NQS:

1. DJM can be given knowledge of when it is to be shutdown, and thus can avoid starting jobs that will not complete in time and instead search for and select jobs that can complete;
2. DJM has precise knowledge of the configuration of the machine, and will only start jobs that can be fitted into the torus, and this knowledge enables fragmentation and resource wastage to be largely avoided;
3. DJM uses a scoring system for job selection, which enables balancing of usage between multiple groups of users to be better maintained thus maximising fairness.

EPCC has investigated DJM but is concerned about long-term support for such a product, and is unwilling to commit to a product that is not necessarily mainstream Cray-supported software.

Suggested Improvements

The configuration changes made since the installation of the system, the efforts made in reordering queues, and the planned efforts to be made in providing user exits to control job selection are only tinkering with the problem.

The linkage between NQS and the MPP is poor, and the following is a list of suggested improvements that might enhance the efficiency of use of a MPP system.

1. Introduce knowledge of planned shutdowns so that premature job termination can be avoided;
2. Enable pools to be linked to specific queues, so that problems that may be caused by starting a 64 processor job in a 256 processor pool and thus prevent the larger jobs from starting, can be avoided, as well as providing concurrent production and development job streams;
3. In addition to checking that there are sufficient processors available, ensure that a MPP job can be placed upon the torus before starting the job;
4. Introduce a sensible interpretation of elapsed

time so as not to penalise jobs with multiple concurrent calls of 'mppexec';

5. Maximise throughput by 'tiling' a fragmented torus with appropriately sized smaller jobs;
6. Consider implementing a relocation scheme to avoid fragmentation.

Results

Usage statistics on the Edinburgh T3D are collected daily for accounting purposes, and are collated on a weekly and a monthly basis.

Within any week, there is a maximum of 53760 processor hours (assuming no downtime) that could be used for user work, and of these a maximum of 48896 PE hours are available for batch work.

The peak utilisation for batch work was during the week 22 May - 29 May when 47239 PE hours were used by batch applications, this corresponding to 96.6% of the theoretical maximum. During that same week, only 2571 PE hours were used for interactive applications. Similar performance was achieved on the week 28 August - 4 September, when 46991 PE hours (96.1% of the possible maximum) were used by batch applications.

Over any weekend (1800 Friday until 1000 Monday), out of a total of 20480 PE hours, 19200 PE hours are available for batch work.

The peak utilisation for batch work over a weekend was the 64-hour period from 1800 hours on 5th May until 1000 hours on 9th May when 19073 PE hours (99.4% of the possible maximum) were used. Over this period, 159 individual mppexec calls were made for batch jobs, with 6 users running 256 processor jobs, 2 running 128 processor jobs and 3 running 64 processor jobs. These users represented 7 different user groups, 5 of which used in excess of 5% of the cycles. Similar results have been obtained in August/September. Such high utilisation is obviously easily achieved if very small numbers of very large jobs are run, it is gratifying to record that it can also be achieved with multiple jobs with fixed maxima residency-times being run on behalf of multiple users from disparate user groups.

It is difficult to define a metric to measure the increase in effective throughput as the workload is constantly evolving. What is clear is that while the total amount of work performed on the machine has remained reasonably constant in the period March - August, the overall efficiency has improved in that the mean turnround for a 256 processor job on the NIGHT queue has been reduced from 8 days to 48 hours. Typically, after a NIGHT configuration, there may be 3 - 4 outstanding 256 processor jobs waiting to run, while earlier in the year there was typically in excess of 12.

Conclusions

It is clear that a number of the limitations discussed above, in particular fragmentation caused by small jobs, the ability for NQS to start a job that cannot be placed within the torus, poor job selection based solely on primitive "first-in-first-out" scheduling and the premature termination of jobs at configuration change time, have created a significant impact on the operation of the system and have caused resources to be wasted.

Experience at Edinburgh has proved that there is a very poor linkage between NQS and the MPP, and that Cray should consider implementing many of the features contained within DJM if it is planning to continue with NQS as the mainsteam product on MPP systems.

The very high levels of processor utilisation on the Edinburgh machine have only been reached by various decisions being made on the acceptable job-profile. The reasons behind these decisions have largely been accepted by the user population, but some of the perceived generality of the system has been removed as resources have been concentrated on supporting the users with the very largest requirements.

There has been an improved balance of resources between production and development use since the upgrade to a non-powers-of-two sized T3D, but true separation of concurrent job streams would

require NQS to be able to target work to a particular pool.

In summary, it has found been necessary to undertake the following to maximise the throughput on the system:

1. Deciding to concentrate almost entirely all production time resources on catering for jobs of the largest and most appropriate sizes (64, 128 and 256 processors)

2. Avoiding the fragmentation problem as far as possible by concentrating on running batch jobs of the appropriate sizes within the 256 processor pool. This process has been much aided by the availability of the additional 64-processors over the original 256 to enable production and development work to be effectively divided into separate pools.

3. Deciding that the consequences of having user jobs aborted prematurely by a configuration change is preferable to leaving resources totally idle.

4. Deciding, in concert with the users, on appropriate job-residency times at different times of the day, and on different days of the week, so as to ensure that a reasonable spread of job sizes and durations may be accommodated without inducing unacceptable job turnround times.

5. Providing 'short' job queues for the principal job sizes to enable proving runs to be undertaken on large numbers of processors, without the requirement for the user to have to endure a long turnaround on a test job and without the fragmentation that may result by allowing some jobs to be run interactively.

6. Maximising fairness by introducing agreed 'rules' with respect the number of jobs that a user may submit, and automatically reordering the jobs within the batch queues so that their use cannot be dominated by particular user groups.

Acknowledgements

This work was undertaken under the terms of the contract between the Engineering and Physical Sciences Research Council and the University of Edinburgh with respect to the operation of the Cray systems at EPCC.

The operational configuration of the Edinburgh T3D has evolved greatly since the original installation of the system in April 1994, and the author would like to acknowledge those who have had significant input into the configuration changes, and the users who have themselves suggested improvements and have accepted all changes to the configuration, and their working practices, with good grace.

Particular thanks are due to Stuart Wilson, the on-site analyst from Cray Research (UK) Ltd for his support, advice and expert knowledge and to Steve Booth (in-depth Scientific Support Manager, EPCC), who has taken a particular interest in the improvement of the batch service for the T3D.

The author is on full-time secondment from the Edinburgh University Computing Services to EPCC and gratefully acknowledges the support of the Director of Computing and Information Technology Services and the Vice-Principal for Academic and Information Services.

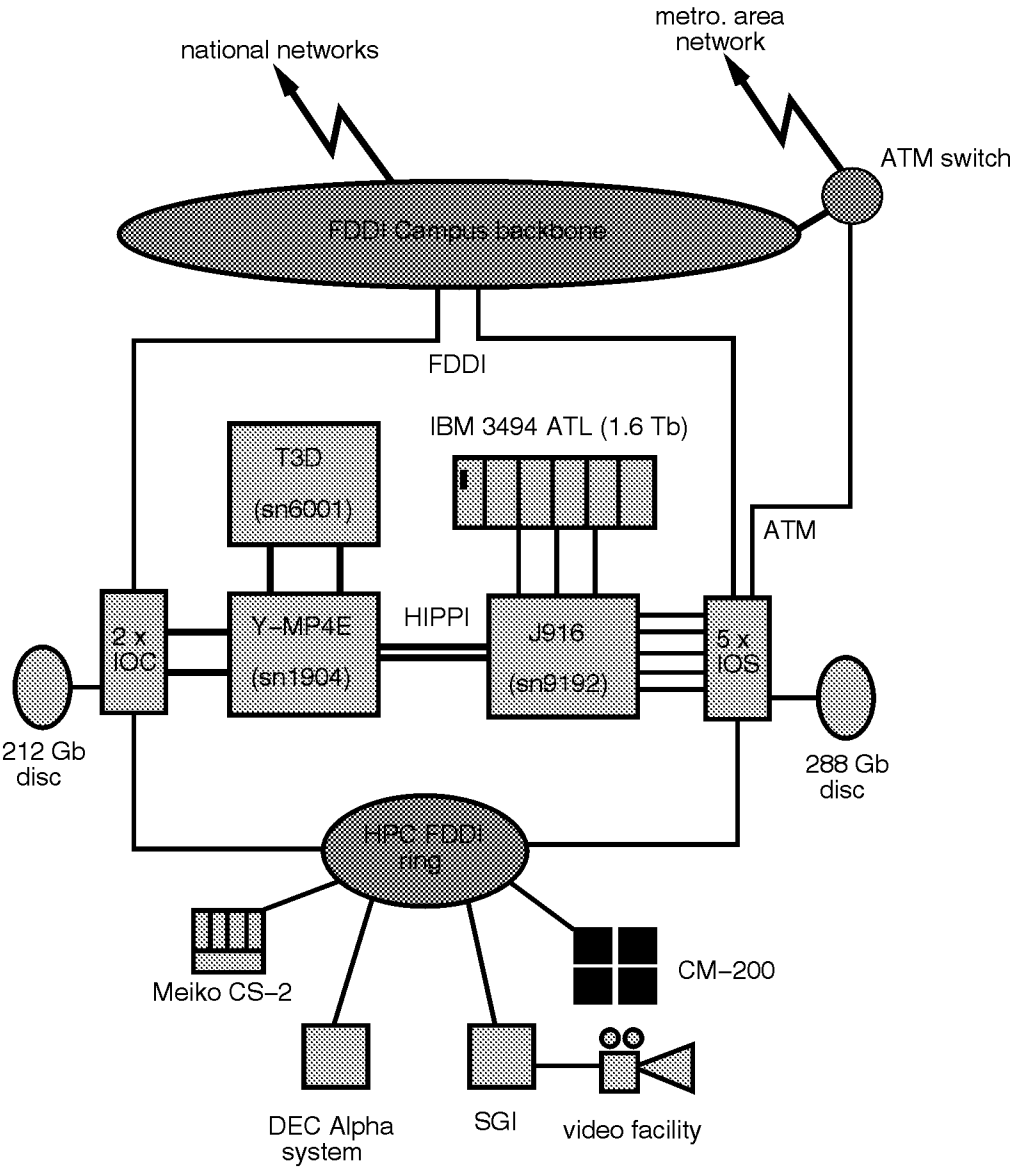
References

[1] Research on the Cray T3D - the First Year, *Engineering and Physical Sciences Research Council, (to be published October 1995)*

[2] Operation of the Cray T3D as a National Facility, *M.W.Brown, Proc. 35th Cray User Group, March 1995*

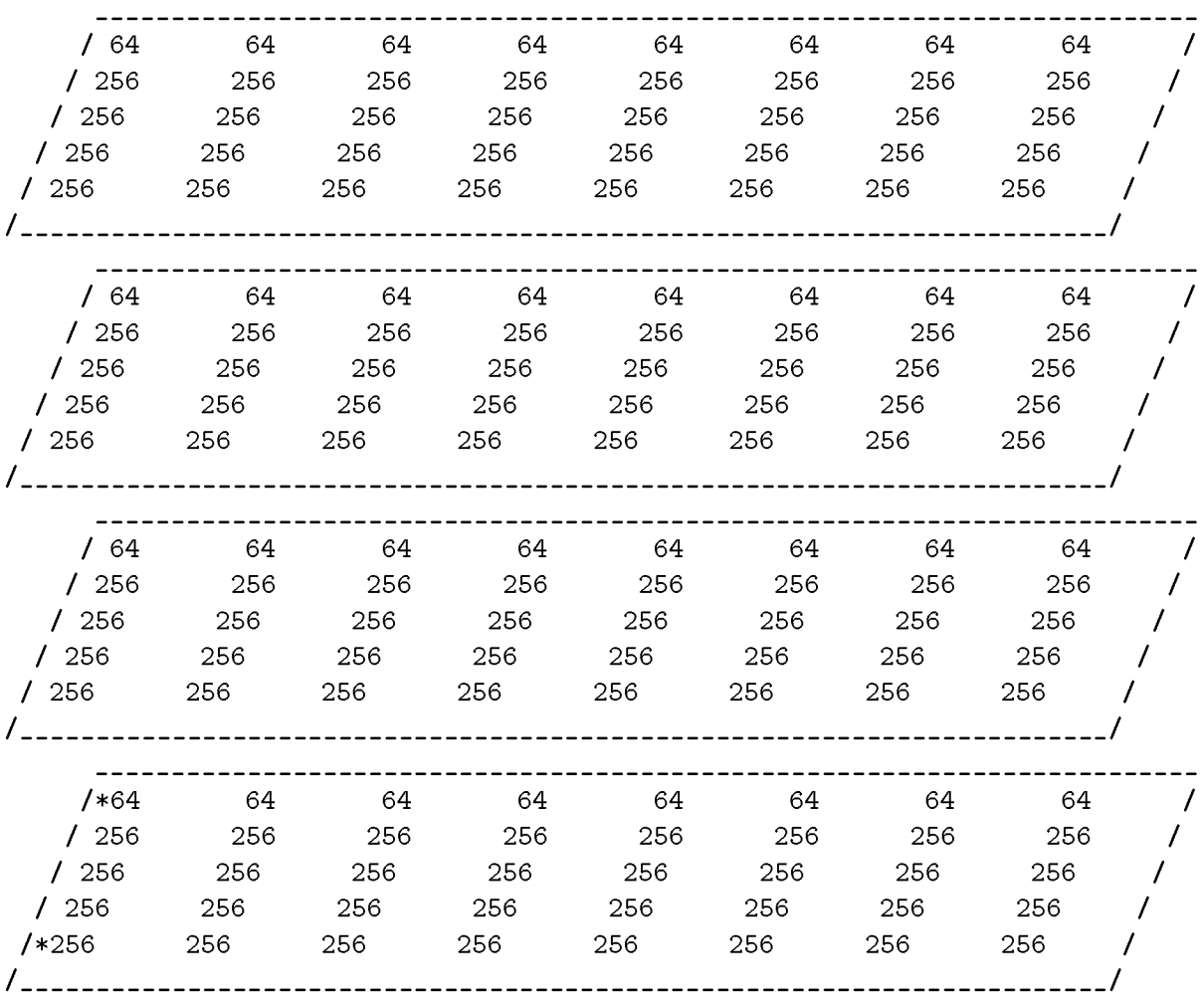
[3] Distributed Job Manager Reference Manual, *Thinking Machines Corporation, 1993*

HPC FACILITIES AT THE UNIVERSITY OF EDINBURGH



October 1995

Pool layout of torus in MCN320 system (in XZ planes)



Schematic representation of the 8 x 4 x 5 torus of logical nodes within the Edinburgh MCN320 system, showing the division into 256 and 64-PE pools.

With the impending upgrade to a MCN384 system, the torus shall have the dimensions 8 x 4 x 6, and a further 'slice' shall appear in the Z direction.

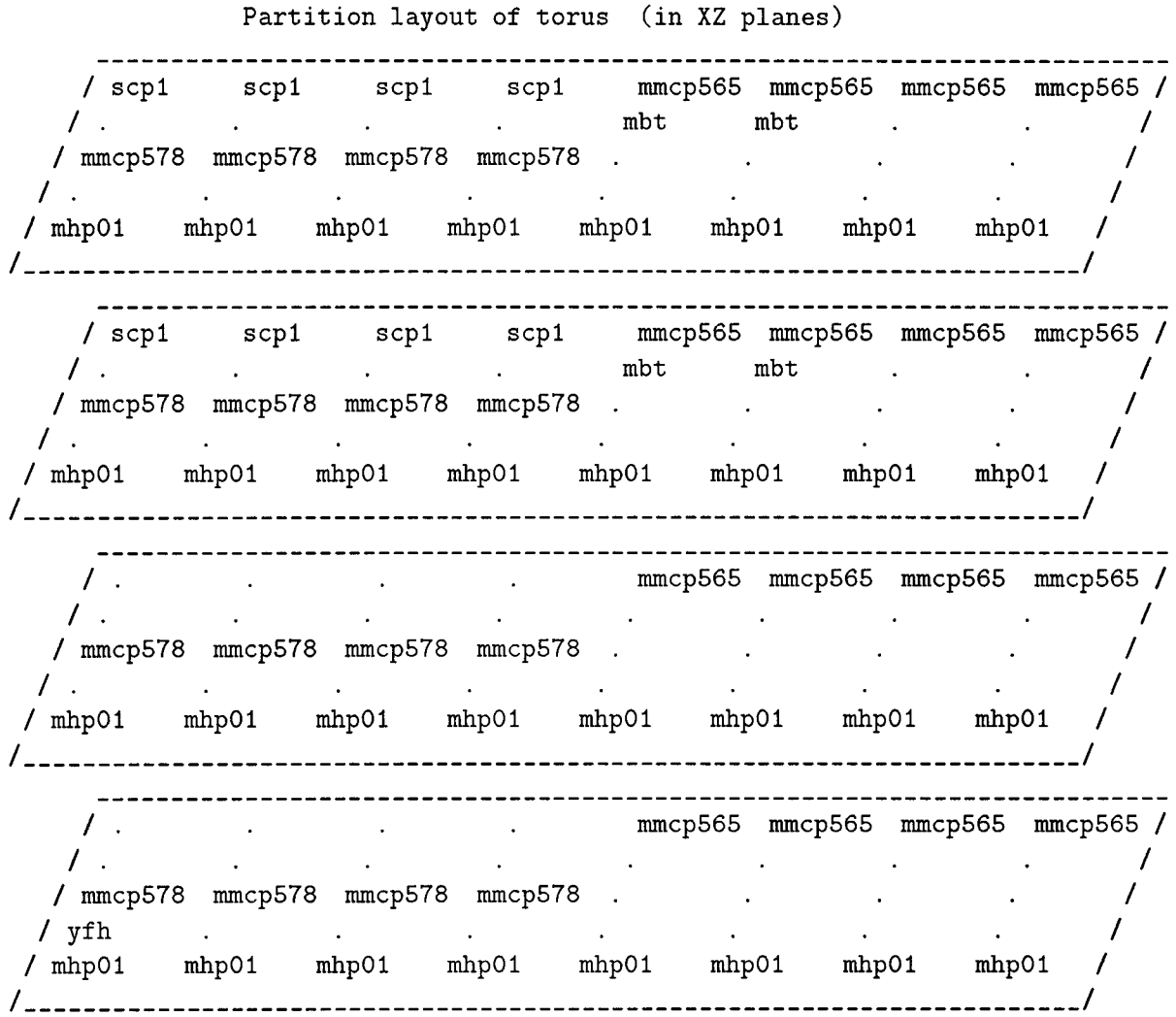
POOL_64 base node: 0x400 (indicated by *64)

POOL_256 base node: 0x000 (indicated by *256)

```
                 Partition layout of torus  (in XZ planes)
          ------------------------------------------------------------------------
         / jpreid   jpreid   jpreid   jpreid   ijb      ijb      ijb      ijb     /
        / .        .        .        .        .        .        .        .      /
       / marobb   marobb   marobb   marobb   marobb   marobb   marobb   marobb  /
      / ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29   /
     / .        .        .        .        .        .        .        .      /
    /------------------------------------------------------------------------/
          ------------------------------------------------------------------------
         / jpreid   jpreid   jpreid   jpreid   ijb      ijb      ijb      ijb     /
        / .        .        .        .        .        .        .        .      /
       / marobb   marobb   marobb   marobb   marobb   marobb   marobb   marobb  /
      / ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29   /
     / .        .        .        .        .        .        .        .      /
    /------------------------------------------------------------------------/
          ------------------------------------------------------------------------
         / jpreid   jpreid   jpreid   jpreid   ijb      ijb      ijb      ijb     /
        / .        .        .        .        .        .        .        .      /
       / marobb   marobb   marobb   marobb   marobb   marobb   marobb   marobb  /
      / ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29   /
     / .        .        .        .        .        .        .        .      /
    /------------------------------------------------------------------------/
          ------------------------------------------------------------------------
         / jpreid   jpreid   jpreid   jpreid   ijb      ijb      ijb      ijb     /
        / .        .        .        .        .        .        .        .      /
       / marobb   marobb   marobb   marobb   marobb   marobb   marobb   marobb  /
      / ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29    ajw29   /
     / .        .        .        .        .        .        .        .      /
    /------------------------------------------------------------------------/

User       PID  Program  State  Pool     Flags  Shape- XYZ   (base)  Elapsed
-------- ------ -------- ------ -------- ------ ------------------- ---------
ijb       29504 a.out    Active POOL_64  I        32= 8x 4x 1(xffe) 00:20:00
ajw29     29540 xfit     Active POOL_256 B        64=16x 4x 1(xffe) 00:19:38
marobb    30727 mmvb_t3d Active POOL_256 B        64=16x 4x 1(xffe) 00:08:03
jpreid    31098 molscat  Active POOL_64  I        32= 8x 4x 1(xffe) 00:04:39
jdg       31128 cetep.DM Wait   POOL_256 Ba~p    128=16x 4x 2       00:05:06
```

Although 128 processors are unused in the 256 processor pool, the necessary XYZ 'shape' of 16 x 4 x 2 cannot be placed. NQS is not aware that the processors cannot be placed, and starts the job which blocks until either of the 64 processor jobs currently running completes.



User	PID	Program	State	Pool	Flags	Shape- XYZ (base)	Elapsed
mmcp565	46736	gbmesonp	Active	POOL_64	I	32= 8x 4x 1(xffe)	00:18:47
mhp01	46799	gamess_m	Active	POOL_256	B	64=16x 4x 1(xffe)	00:18:17
mmcp578	48232	gbmesonp	Active	POOL_256	I	32= 8x 4x 1(xffe)	00:07:00
scp1	48750	DL_POLY	Active	POOL_64	I	16= 8x 2x 1(xffe)	00:02:44
mbt	48937	dr	Active	POOL_256	B	8= 4x 2x 1(xffe)	00:01:28
yfh	49082	conn	Active	POOL_64	I	2= 2x 1x 1(xffe)	00:00:39
cgw	49144	full_noi	Wait	POOL_256	Baq	64=16x 4x 1	00:00:11

The machine is badly fragmented. A 32-processor interactive job has started within the 256 processor pool as resources were available there when the 64 processor pool had been fully utilised, followed by a 2 processor job. An 8 processor batch job has started, followed by a 64 processor job which has blocked as it cannot be placed (even though 150 processors remain unused in the 256 processor pool)

With manual rolling, the systems administrator could move the 2 processor job to somewhere within the 64 processors pool, and thus enable the blocked 64 processor job to start.