THE MULTITASKING SPECTRAL MODEL AT ECMWF

David Dent

European Centre for Medium Range Weather Forecasts
Reading, UK

1. INTRODUCTION

ECMWF has 3 major applications which could benefit from multitasking because they are both time critical and require a large proportion of main memory. To date, only the forecast has been modified to make use of more than one processor, although work is proceeding on the other applications. This paper discusses the spectral model in its state of development at the end of 1984.

2. HISTORY

The original ECMWF production forecast was made using a grid point model. Over a period of 2 years, a spectral model was developed to replace it. This went into daily production in April 1983 executing on a CRAY-1A, with spectral resolution T63.

The code is independent of resolution and can be run without recompilation using any desired data. It consists of:

96000 source lines

26000 Fortran statements

Since the resolution T63 was chosen as appropriate for a CRAY-1, a computationally more demanding resolution is possible on a CRAY-XMP. Given the available configuration, a resolution of T106 has been chosen for a comprehensive set of meteorological experiments with the target of making this model available for production use by April 1985.

To achieve an acceptable wall clock execution time, it is obviously essential to make efficient use of both central processors of the CRAY-XMP from within the application code. Hence, a multitasking version of the spectral model has been developed over a period of about 1 year. The first working version went into regular experimental use in July 1984. Since then, efforts have been made to reduce the execution time by identifying and removing inefficiencies.

3. USE OF MAIN MEMORY

The model is unusual in that it makes extensive use of a locally designed memory manager to provide easy and safe organisation of main memory for the various sub-processes within the code.

Since the model was developed and will continue to be developed by a team of scientists, it is extremely useful and productive for any member of the team to be able to allocate a portion of memory for his own particular needs, safe in the knowledge that his allocated space will not be used by any other member of the team.

Use of the manager depends on the POINTER statement, which is an extension to Fortran 77 supported by the Cray Fortran compiler. It allows an array to be dynamically dimensioned, since the array dimension may be a variable in COMMON storage and hence known only at run time. Memory is allocated from a block obtained at initialisation time from the Cray heap manager. An allocated array is identified uniquely by a character name and an integer code. Its base address may subsequently be obtained by another subroutine through use of a LOCATE routine. An UNLOC routine is available to return the space when no longer required.

```
      POINTER (IPT , DIV(ND) )


      CALL ALLOCA(IPT , LENGTH , NAME , KODE )

      CALL LOCATE(IPT , NAME , KODE )

      CALL UNLOC (NAME , KODE )
```

where: DIV is the name of the array of length ND addressed through
       the pointer IPT

```
        NAME is the character name   ) which together identify
        KODE is the integer code     ) the allocated space
```

In multitasking mode, a subroutine commonly executes in both processors simultaneously. Since the subroutine often requires array space to hold the results of calculations, this output space must be unique for each subtask. This can be achieved by using a locally dimensioned array in the usual way, but this space is obtained from the Fortran controlled stack and is released at the end of the subroutine. By using a different integer code value when allocating an array, the memory manager creates a unique space for each execution of the routine and this space remains available for other subsequent routines to locate until explicitly released.

Such a convenience to the programmer of course costs some execution time which must be weighed against the scientists productivity and the ease of code maintenance and development. The overhead in the spectral model is approximately 5% of execution time.

4. ECMWF CRAY XMP CONFIGURATION

From the point of view of the spectral model, the principal characteristics of the Cray-X2200 installed at ECMWF are-

2 Central Processors

2 Mwords of central memory

16 banks of memory

16 Mwords of Solid State storage device (SSD)

80 Mwords/sec memory to SSD transfer rate

5. COMPUTER RESOURCES USED BY THE SPECTRAL MODEL

At resolution T106, the single-tasking model requires:

1.5 Mwords of central memory

15.3 Mwords of SSD

There are 3 major work files:

Legendre coefficients - 950 KW - read twice each step

grid point data - 8.7 MW - read and written each step

fourier coefficients - 5.7 MW - read and written each step

total 15.3 MW - 30 MW I/O per step

Putting files on a device with such a high transfer rate to/from central memory allows I/O to be carried out synchronously without much overhead. This reduces the central memory requirements for buffer space and costs less than 4% of the elapsed time for a 10 day forecast.

6. MULTI-TASKING INTERFACE

The following facilities available in the Cray multi-tasking library are used in the model:

```
CALL TSKSTART(ctltab,routine)
CALL TSKWAIT (ctltab)
CALL LOCKON  (lock)
CALL LOCKOFF (lock)
```

where- 'ctltab' is a task control block
'lock' is a unique lock identifier
'routine' is the name of a subroutine to be executed

These tools enable tasks to be started and synchronised, and critical areas of code to be protected against simultaneous execution. Event setting is also supported in the library but the current version of the model does not use this technique. It is possible to pass parameters to 'routine' but this facility is also not used.

7. GENERAL STRUCTURE

The model is organised into 2 scans over the data as shown in figure 1. Within each scan, there is a loop over all latitude rows (160 for the T106 resolution). Between scans is a smaller area of computation associated with diffusion and semi-implicit calculations. The loop over time steps is repeated 1200 times for a 10 day forecast. However, every 16 steps, significant additional computation is performed by radiation calculations.

Within Scan 1, the following are the principal components of work-

I/O: read Fourier coefficients
I/O: read legendre coefficients
I/O: read and write grid point data
computations in Fourier space
FFT - grid point computations - FFT
semi-implicit computations
computations in Fourier space
compute direct legendre transforms

Within Scan 2, the main items are-

I/O: read legendre coefficients
I/O: write Fourier coefficients
compute inverse legendre transforms

A multi-tasking version of an application requires more main memory than its single-tasking equivalent. Given (a) the desire to maximise the resolution and (b) the shortage of main memory, it is important to select a multitasking strategy which has low memory requirements.

It turns out to be convenient and efficient in memory to split Scan 1 and perform it in 2 pairs of subtasks with a synchronising point in between. This is because each Northern row generates the symmetric part of a Fourier component while the equivalent antisymmetric part is generated by the appropriate Southern row. Both components are combined in different ways to provide contributions to the legendre transform. By computing one Northern row and one Southern row simultaneously, not only is the memory requirement minimised, but also the legendre computation is performed efficiently.

Part of the diffusion calculation is also multi-tasked and Scan 2 can be computed 2 rows at a time (see figure 2).

There remain some relatively small parts of the code which are computed in single-tasking mode.

The memory requirements for this multi-tasking strategy are 1.8 Mwords. Note that alternative strategies are of course possible. However, subtask structures which may be preferred for optimising reasons require either more central memory or additional SSD.

8. OVERALL TIMINGS

All the timings reported here are elapsed times corresponding either to a single time step or to a complete 10 day forecast.

For a normal timestep:

single tasking : 25.36 seconds/step
multi tasking : 14.28 seconds/step
speedup ratio : 1.78

For a radiation timestep:

single tasking	:	75.0	seconds/step
multi tasking	:	39.4	seconds/step
speedup ratio	:	1.9	

These times correspond to a total time of 6 hours for a 10 day forecast, including the creation and post-processing of history data.

The Cray-XMP has the capability of simultaneously reading and writing to memory from the vector registers. This feature may be switched on or off by means of a simple control statement. The above times were measured with bidirectional transfer enabled. The following times were measured when bidirectional transfer was disabled:

single tasking	:	26.71	seconds/step
multi tasking	:	14.21	seconds/step
speedup ratio	:	1.88	

Thus, for single tasking, switching on the bidirectional mode speeds up the model execution by about 5%. However, when the model is multi-tasked, there is no corresponding improvement. This is easily explained since a 16 bank memory can at best reference only 4 words per clock period. With both central processors referencing memory at the maximum rate of 2 vector reads and one vector write, the code is trying to reference a maximum of 6 words every clock period and is therefore slowed down. If the same multi-tasked model were to be run on a Cray-X2200 with 32 banks of memory, an estimated saving of 15 minutes for a 10 day forecast would be achieved.

9. MORE DETAILED TIMINGS

Since the above timings are very simple and made at the very highest level they tell nothing about the behaviour of individual tasks within the model. Currently, there is no support within the Cray multi-tasking library for obtaining detailed timings. Consequently, all the following timings were obtained by inserting code into the model at strategic places in order to record times as reported by the real time clock. The measurements were done in such a way as to disturb the model as little as possible. The model was run in a dedicated environment with no disturbances other than any caused by the operating system (COS X.13). Analysis of the measurements was done subsequently in a normal batch environment.

The average times taken by each of the tasks as identified in the previous section are shown in Fig. 3.

By measuring the time taken by the Cray multi-tasking library routines, it is possible to obtain estimates of the cost of starting tasks etc.

For TSKSTART , three distinctly different times are observed as follows:

40 milliseconds	for one case only
0.4 milliseconds	for 96% of all TSKSTARTs
0.04 milliseconds	for 4% of all TSKSTARTs

The expensive start corresponds to the very first TSKSTART in the complete application, when additional memory has to be requested from the operating system for table space.

The intermediate time corresponds to the case when a 'logical CPU' has to be established (table creation etc).

The shortest time corresponds to the case when a logical CPU already exists. In this execution, the Cray multi-tasking scheduler has released the logical CPU in nearly all cases before the next task is created. The small percentage of fast TSKSTART times were all observed for TASK 2 where there is a very small time gap after completion of TASK 1. In the future it will be possible to tune the scheduler to retain the logical CPU in all cases.

The measured minimum times for other multi-tasking calls are:

TSKWAIT	0.007 milliseconds
LOCKON/LOCKOFF	0.001 milliseconds

Hence it is clear that the TSKSTART times dominate the task overheads.

The approximate total overhead cost in a 10 day forecast is:

3 * 80 * 1200 * 0.4 milliseconds

which is about 2 minutes or 0.7% of the total time

With scheduler tuning, this is likely to be reduced to 0.1%.

An obvious conclusion is that task overheads are small compared to the size of tasks which exist in the spectral model.

10. INEFFICIENCIES

By measuring the amount of time spent outside of the tasks, it can be seen how much of the code has been multi-tasked and therefore what additional improvements might be made in the future (see Fig. 4).

The TSKWAIT time reported in the previous section was the minimum observed i.e. for the case where the master task completed after the started task and was therefore not held up in the synchronising process. By examining average TSKWAIT times, it is possible to obtain estimates of how imbalanced the pairs of tasks are. Figure 4 shows that these imbalances account for about 4% of the overall model time. Most of the imbalance was observed in TASK1. TASK 2 and TASK 3 imbalances were smaller by a factor of 9.

There are at least 2 reasons for this imbalance. One concerns LOCKS and will be discussed below. The other concerns the nature of the computation in grid-point space (part of TASK 1). Although the amount of work done for each latitude line is exactly equal for the dynamics part of the code, this is not always true in parts of the physical parameterisation. Convection and condensation calculations are affected by synoptic conditions and will therefore vary in space and time. The magnitude of these variations in terms of computing expense has not yet been measured.

LOCKS are used to protect critical regions of code in some 20 places, mostly for statistic gathering purposes. These locks all occur in TASK 1 and are mostly insignificant in time. However some random I/O is carried out to a single dataset which is common to both tasks. In the current Cray software, a lock is applied whenever I/O is initiated to any dataset, so that the strategy of splitting this dataset into 2 will not be useful until this high level lock is moved to the level of the dataset. Indications are that this causes most of the imbalance observed in TASK 1.

11. FUTURE IMPROVEMENTS

Since the target time for a 10 day forecast is approximately 5 hours, there remains substantial optimising to be done before the spectral model is fast enough for operational use. However, significant improvements have already been made (see Fig. 5). By reducing the single-tasking time and by attacking the out of balance inefficiency, it should be possible to improve the multi-tasking performance. It may be possible to bypass the I/O lock and hence substantially reduce the imbalance due to locking. There is also some scope for optimising at the loop level (loop unrolling etc).

Alternative multi-tasking trategies can be tried in order to reduce the number of synchronising points and hence the imbalance time. Unfortunately there is little scope for this effort given the constraints of central memory and SSD space. EVENT synchronising is known to be more efficient than TSKSTART-TSKWAIT and this could be implimented easily in at least part of the application. However, since the task overhead is relatively small, this is unlikely to be useful.

It is interesting to speculate on the model's performance when executed on future hardware with additional processors. Code already exists in the model for N processors but it is largely untested to date. It is based on the current multi-tasking strategy and therefore performance estimates may be made based on the measurements reported earlier in this paper. Fig. 6

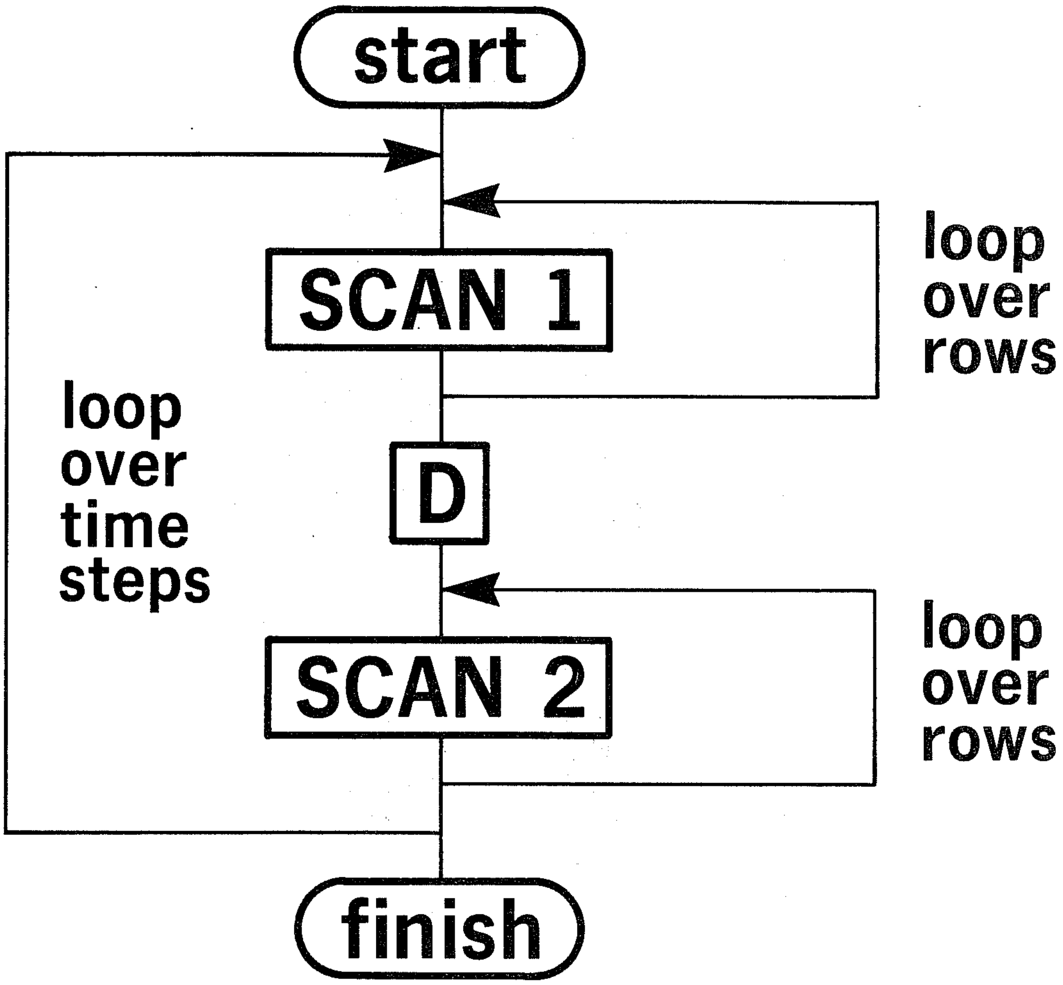
indicates that a multi-tasking efficiency of about 3.2 could be achieved with 4 processors, but with 16 processors a speedup of only 8.2 would be achieved over the single-tasked model.

12. SUMMARY

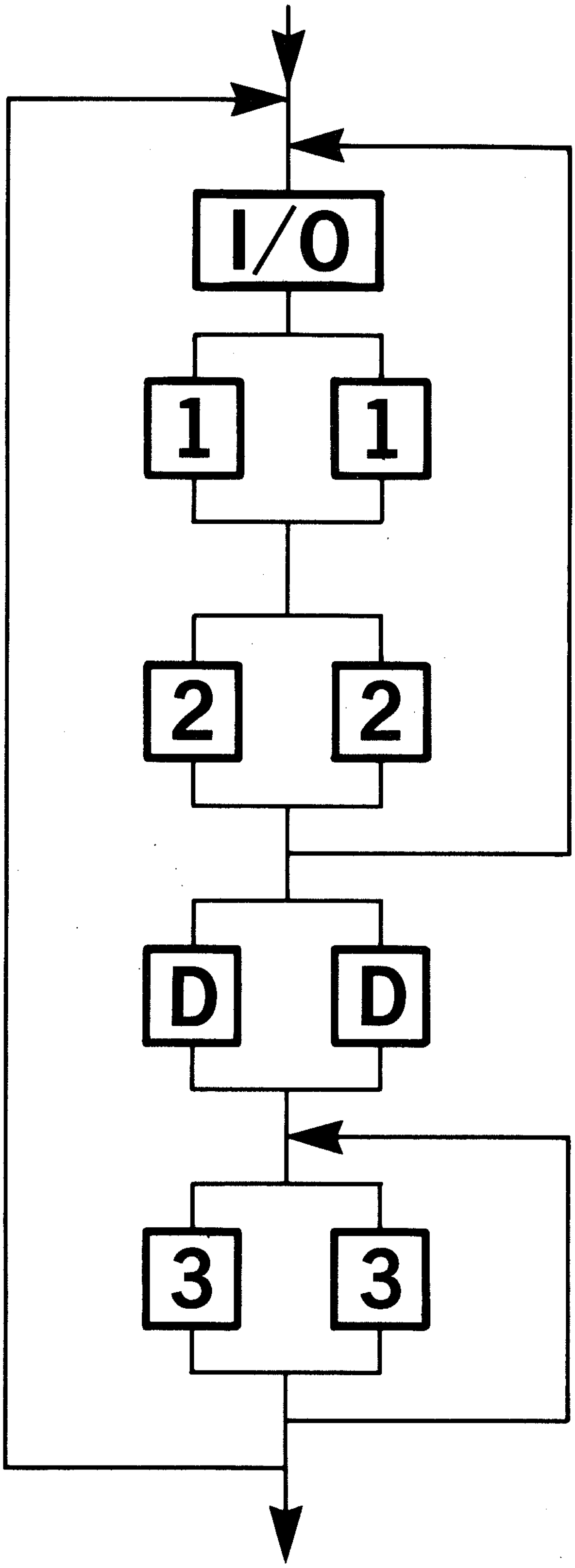
The ECMWF spectral model is a large application which has been successfully adapted to a multi-tasking environment. The task overheads are small compared to the task sizes. Improvements must be made in the elapsed time before the model can be used operationally, but these appear to be achievable in the time frame required.

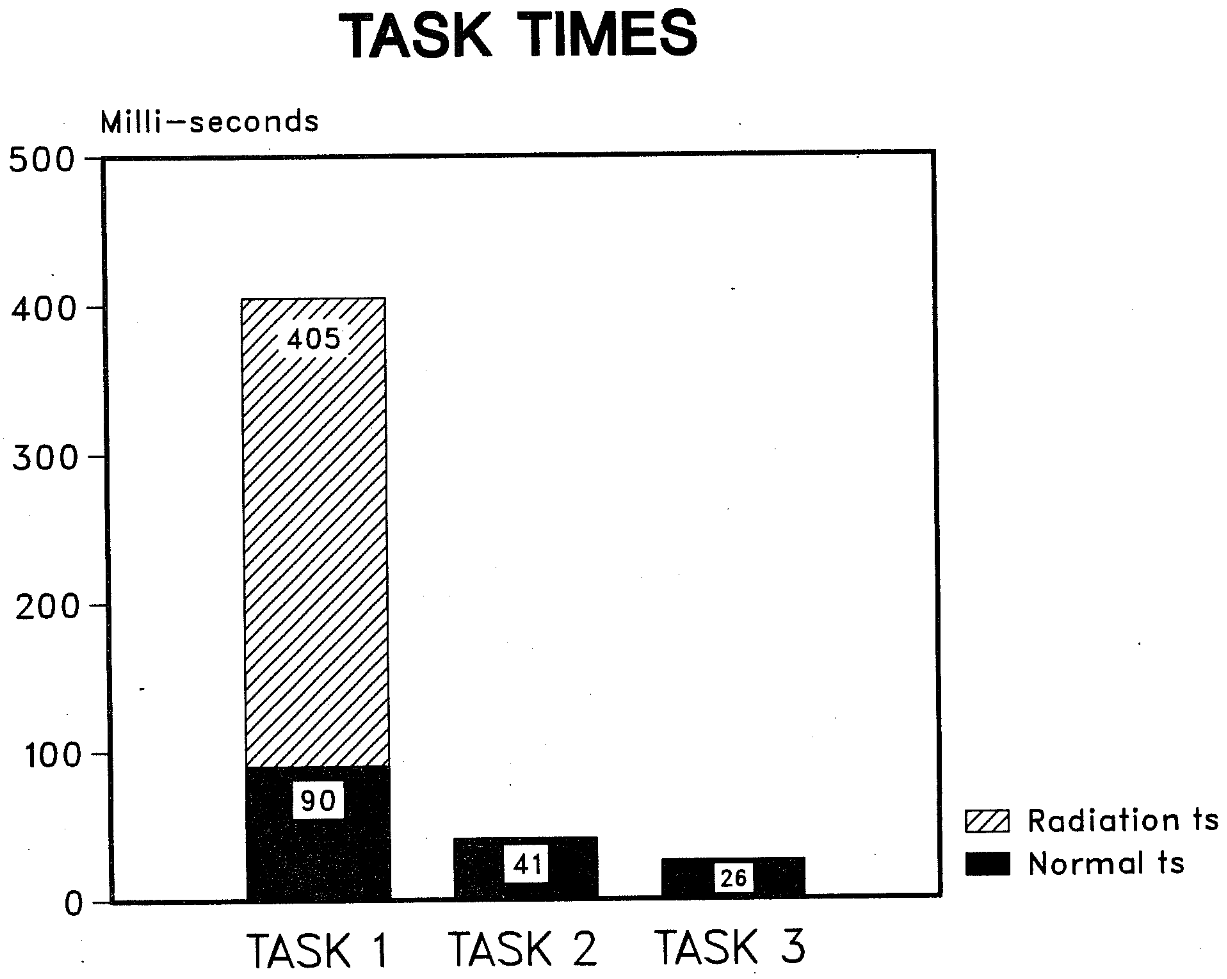
The code provides the basis for execution on future machines but in order to make good use of increased number of processors some refinement of the multi-tasking strategy is necessary.

GENERAL STRUCTURE

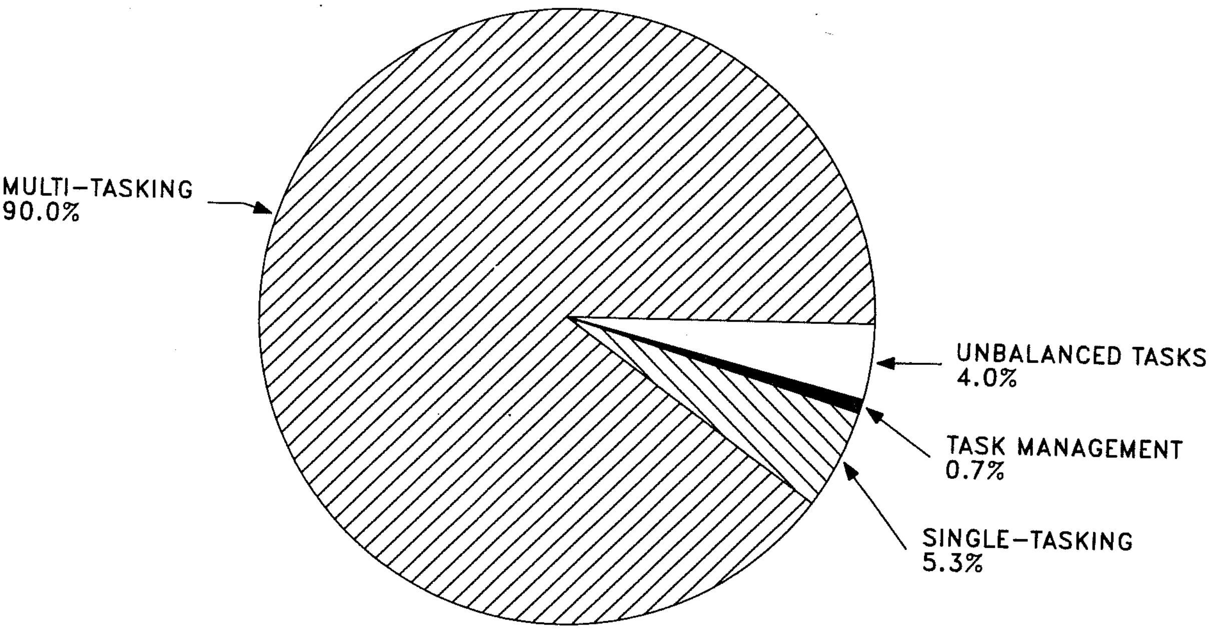


MULTI-TASKING STRUCTURE

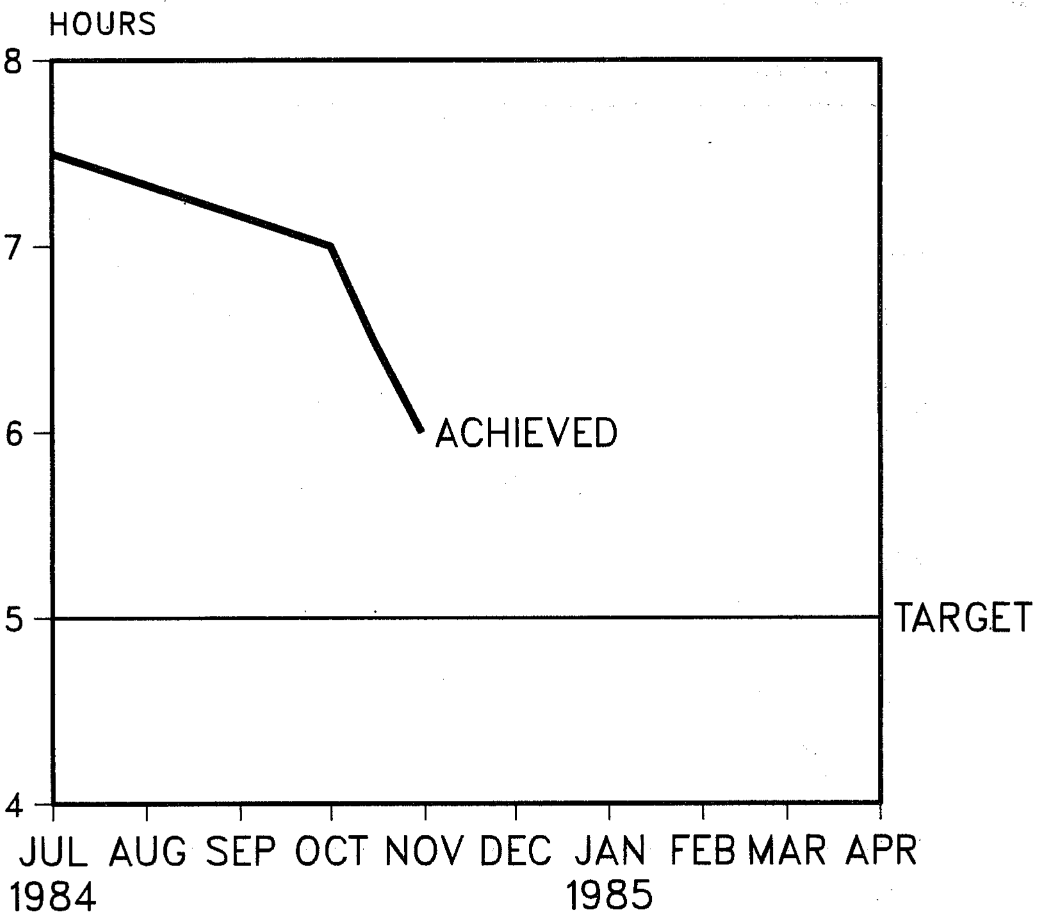



TASK TIMES
Milli-seconds
500
400
300
200
100
0
405
90
41
26
TASK 1
TASK 2
TASK 3
▨ Radiation ts
■ Normal ts

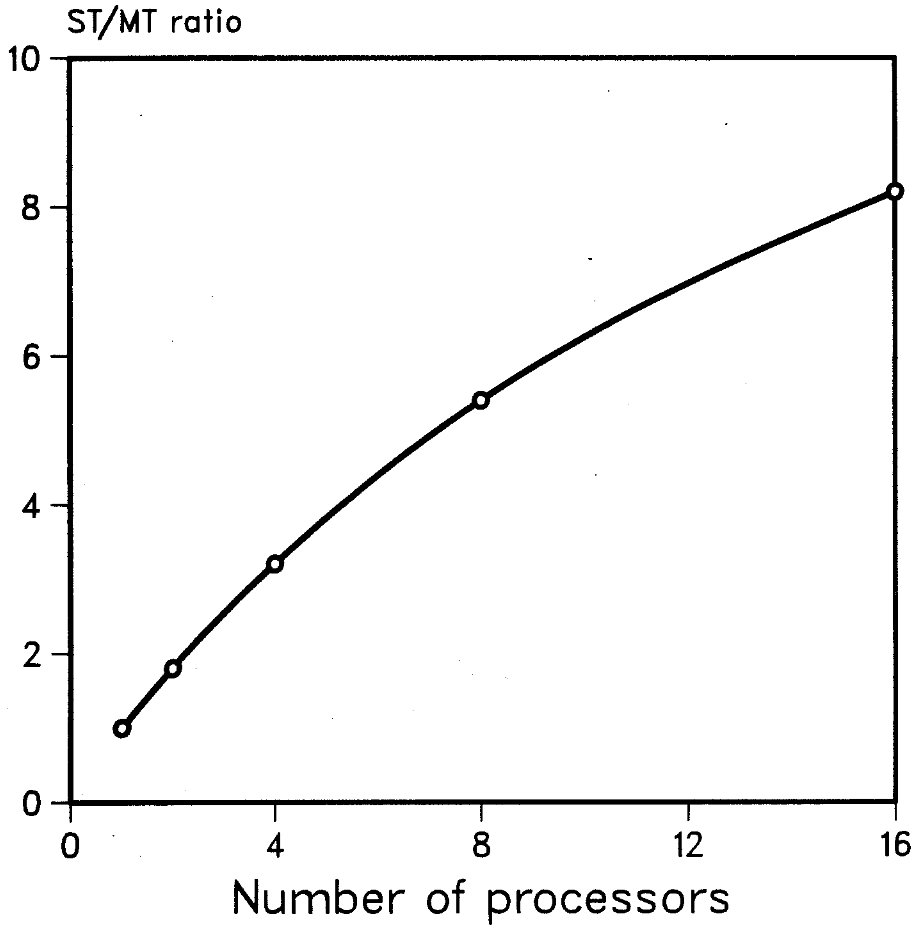
MULTI-TASKING INEFFICIENCES



FORECAST OVERALL TIME



EFFICIENCY WITH MORE PROCESSORS


ST/MT ratio
Number of processors