The scheduling problem, part II

mjk, 23 oct. 84

Some reconsiderations of the problem indicate that the automatic mode suggested earlier (see attached note) may be more ambitious than warranted.

The problem is this: the astronomer needs to be given a start/stop time before he generates a program, since otherwise he cannot be sure that all his sources are above the horizon. Thus a schedule must already have been established. Some software to simplify this scheduling task may well be useful, but can probably not be left in an automatic mode.

I therefore now favor this somewhat simpler, even if more labor intensive, scheme:

1. A preliminary schedule would be drawn up by the time allocation, scheduling , or whatever committee. Each astronomer would be advised of his time slot, and then expected to run the task SCHED to produce his observing program.

2. After an array reconfiguration, a calibration run would be done by the resident array manager in order to calibrate the array. (And to load the ARRAY CONFIGURATION parameters table).

3. The array manager would then make up the definitive schedule - essentially the preliminary schedule. It would take the form of a file, with an ordered list of programs, and would probably cover a period of about a week. The task OBS would refer exclusively to this file, working its way through it. The task OPER would be able to add and delete entries in this file. Should major hardware changes indicate that some programs are no longer viable, then a fresh schedule would be made.

It would probably be advisable to have the schedule file permanently on display - a dedicated terminal, perhaps. This display would be continually updated, showing the current status - eg. program completed, underway, skipped, etc. A hard copy of this file would be useful for archival purposes, as well as a guide for subsequent scheduling.

The REQUESTS file of earlier notes will no longer play a role insofar as an automatic mode is concerned; it will still be needed in order to keep track of long term (multi-day) programs. The SCHEDULE file will have the same

The scheduling problem

mjk, 16 oct. 1984

In essence we have to translate the astronomers' requests into an ordered set of instructions to OBS, the task which directs the collection of data.

In its simplest form the astronomer's request is for a time allocation.

The array manager needs to select from the set of requests those consistent with the current hardware availability and array configuration; this subset will then need to be ordered so as to minimize idle time.

Elaborations.

1. An astronomer may request a multi-day configuration. But there are, for example, a number of 4-day 6 km. configurations. The scheduling criteria will thus depend on whether or not any time has already been allocated: any one of the different sets would be suitable if no time had yet been granted, whereas only one specific subset may be acceptable if the observing has already started. The astronomer should need only to submit one request, (eg, for a 4-day, 6 km array); the scheduling software should look to the details. In particular, some care will be needed to ensure that the bookkeeping is efficient and uptodate.

2. The array manager will need assistance in judging when and how to reconfigure the array. In principle these decisions should be made well ahead of time, but short term considerations may enter; for example, the hardware availability may change, or weather conditions adverse to high frequency observations may set in.

3. It should be easy for the array operator/manager to stop the current data collection and restart on a new schedule - ie. the final stages of the scheduling should be simple.

Some definitions

record format, but will be the file accessed by OBS.

4. Array reconfiguration time would also be a convenient time to update the various files: PROGRAM files could be deleted, entries removed from the REQUESTS file, the SCHEDULE file archived, and the parameter files updated on the basis of the previous calibration observations.

5. The REQUESTS file should probably contain all programs submitted; the SCHEDULE file should be specific to a process (Compact array, LBA observe, or LBA playback).

An astronomer will request that a PROGRAM be observed, where a PROGRAM is a collection of SCANS. That is,

a SCAN is an uninterrupted set of data samples of a particular position on the sky. It is characterized by a start and stop time, and right ascension, declination, frequency bands and array configuration.

a PROGRAM consists of a set of scans, all at the same frequency and array configuration. In general a PROGRAM will be geared to a particular object; the scans will collect the data on the object, and also the relevant calibration sources.

A possible scenario.

a. The astronomer will submit a program REQUEST: he will run the scheduler task to generate a file which contains a number of CARDS, each CARD containing all the data necessary to run a scan. (A CARD is the physical representation of a SCAN). The file of CARDS is called the PROGRAM file. It will be a text file; its name will be that of the associated program. The scheduler will also enter the program name in the REQUESTS file.

b. The collection of data - appropriate driving of the correlator and telescopes, etc, will be done by OBS. The selection of the next program to run will be done by the task SELECT, of which there will be one for each OBS.

c. SELECT will generally be dormant; it will be activated by a prompt from OBS ("give me the next program"). SELECT would then scan the REQUESTS file, determine which are consistent with the current hardware availability and array configuration, and choose the next suitable program.

d. OBS should have 2 modes of operation:

- automatic, with SELECT providing the next program upon completion of the previous program;

- manual drive, with OPER overriding: OPER could issue to OBS the command NEXT= program file name.

OBS will be responsible for the bookkeeping - the updating of the REQUESTS file.

e. OPER should be able, through SELECT, to examine the

REQUESTS file.

The operator may wish to know which program will be run on completion of the current program; he may also wish to know how many programs are left needing the present array. Conceivably, he may be able to determine which would be the best next array configuration.