Proposed Requirements Relating to Software "Latency"

1) Efficiency during the operations phase.

Here it can be assumed that the system is operating in a fully pre-planned mode – that is to say it is executing a set of Observing Blocks which have been prepared earlier. Clearly the goal is to maximize the fraction of the time that the antennas spend integrating on-sky. It is not practical to set a single target for what this fraction should be because what is possible depends strongly on the nature of the observation: some experiments involving just the detection of faint sources may use long integration times and require very little calibration, so the fraction of time on-sky should be very high; others projects, where the goal is high accuracy on many bright sources, will require lots of calibration and many slews of the antennas, so the on-sky fraction will be lower. This brings out the fact that, even if we had software that operated instantaneously, there would still be some overheads associated with the hardware – slewing the antennas, moving the calibration loads, etc. A great deal of money and effort has gone into keeping these hardware overheads low, so the requirement on the software is phrased in the following manner: *over the full range of typical observing programs, the amount of time lost due to the software overheads should be "modest" compared to the time that is inevitably lost due to the hardware limitations*. The term modest is used here because one the one hand it is not realistic to demand that the software overheads are completely negligible, but on the other hand we should not allow them to be the dominant source of overhead. To make this explicit, consider an observing block where the required time integrating on-sky is *t_on* (this time should include the integration time on calibration sources). With a model that accounts for things like the slewing speed of the antennas and time to move the calibration loads one can work out the total time *t_total_min* that it would take to execute the block if there were no software overheads, so the minimum possible overhead is *over_hard = t_total_min – t_on.* If the actual time taken to do the observation is found to be *t_total_act*, then the real overhead is *over_hard+soft = t_total_act – t_on.* In some cases the software can and should be doing things while waiting for the hardware, but in other cases the software execution will be the holding item. The increase in the amount of overhead due to the software is given by the ratio *R* = *over_hard+soft* / *over_hard.* The proposed goal for this increase is 25%, i.e. *we want the value of R averaged over the range of observing programs to be no greater that 1.25.*

If we adopt this as the overall target, then we can proceed to construct a "latency budget" for the various individual steps involved – time to start and end a scan, time to apply a calibration, time spent writing out data, etc. To analyze this we need a set of benchmark observations. Four or five of these ought to be sufficient to cover the range of cases. Some suggestions are as follows:

1. Weak Source Detection, i.e. long integrations, minimum calibration.
2. Snapshots on bright sources using fast-switching, i.e. short integrations, lots of calibration.
3. Mosaic of a molecular cloud.
4. Molecular line survey of a field.

To these must be added some "observatory" functions, of which the most obvious are baseline determinations and keeping track of the fluxes of calibration sources. In addition we need models of some typical single-dish observations: on-the-fly maps, position-switched and perhaps frequency-switched observations and individual pointings with the nutator running.

We also need to define the performance that we are assuming for the hardware. At the end of this note I have made a start on listing those. The next step is to put together a spreadsheet to calculate the ratio *R* for each case given a set of plausible times for the software operations.

2) Efficiency for Commissioning Activities.

Here the considerations are rather different, partly because we cannot assume that we are operating in a pre-planned manner, but also because we are trying out untested functions so we have to expect that there will be crashes and other problems. Note that activities of this sort will continue right into the operations phase, so this is not just a short-term consideration. Here the time spent integrating on sky is not the key point. Instead the obvious points are:

1. The response to commands. In general the time between the user hitting the return key and things starting to happen in the hardware should be *no longer than 5 seconds, with 2 seconds as a goal*.
2. Time for windows with displays to open and results to appear after the measurement has been completed – again *no longer than 5 seconds with 2 seconds as a goal*. Note that longer waits may be reasonable in some cases, e.g. if a dirty image has to be produced.
3. Time to restart after a crash. This should include the shutting-down process as well as the start-up. There is an existing system-level requirement to be able to restart the whole system in 15 minutes ("tbc"!). As far as I know there is no budget for this 15 minutes but it obviously should include some time to get the hardware set up. The time taken to restart the software should therefore have and *absolute upper limit of 15 minutes and the proposal is that the goal should be 5 minutes*.

An important point is that there should always be an indicator that the system is "thinking" and so the user should wait as patiently as they can rather than starting to push more buttons.

3) AIV testing.

The emphasis here is on carrying out a series of standard scripts to test various aspects of the antenna performance and the functions of other systems. At present these include holography, pointing tests with the optical telescope, beam maps on planets, sky-dips plus various receiver stability tests, etc. Future items would be holography on astronomical sources and polarization observations. In all cases the requirement is that the execution of these functions should be "reasonably" efficient – i.e. the *software overheads should not take up more than a small fraction of the total execution time – e.g. 10 or 15%, perhaps 25% in exceptional cases*.

Notes on HARDWARE execution times:

1. Antenna drives: 6 deg/sec in Az, 3 deg/sec in El. Settling time, say 2 seconds. Take the "typical" distance to the next source to be 72 degrees in azimuth and 36 degrees in elevation, so the time for a slew is 12 seconds at full speed plus 1 for starting and stopping and 2 seconds for settling giving 15 seconds total. For moves to and from an amplitude calibrator assume 12 degrees in Az or 6 in El, so 5 seconds total. For switching to a phase calibrator take the specified 1.5 seconds. Pointing observations (which we can assume are done with a "five-point") are somewhat special – the position steps are small but we do have to be sure that the antenna position has settled well. Take 2 seconds for this and say 3 seconds for the move to the pointing source and another 3 for the move back so the total time taken in moves for a five-point is 2 times 3 plus 4 times 2 equals 14 seconds. For interferometric pointing we may need to do half the dishes at a time, in which case the total goes up to 22 seconds.

2. Calibration device: specified time for the hardware moves to do a hot and ambient laod measurement was originally 2 seconds but this was changed (unilaterally) to 9 sec. The present design actually gives about 7.5 sec. Adopt this value.
3. Local Oscillator tuning. The specification for tuning both within a band and between bands is 1.5 seconds (although note that there is a "wish" in the System Specs to do it in 0.1 sec for phase referencing). There is however an overhead of 20 seconds for setting up a Laser Synthesizer. It would be reasonable to assume that this is needed at the start of an observing block but that it will be possible to avoid this during rest of the block by planning ahead. There are 2 sets of lasers in each synthesizer specifically to allow for this pre tuning. Also note that the Line Length Correctors may need a reset once per hour, but this may only take ~1 second(?), and that the polarization may need to be realigned once per 12 hours – 1 minute for this (?).
4. Front end selection and set-up. In principle there are no additional overheads here that I am aware of. There are no moving parts and switching of IF and LO connections, together with readjustments to bias voltages, etc., should all be covered in the 1.5 seconds already allocated for LO tuning. An issue to be checked is the warm-up time when a new band is being brought into play. The Front End IPT has allowed themselves 15 minutes for this. If this is really necessary then the scheduler algorithm will have to take this into account and there will need to be a mechanism for getting the right bands turned on in advance. (Up to three bands can be turned on at any time.)
5. Back-end antenna article set-up. Again there should in principle be no additional overheads here although at present we are taking up quite a lot of time measuring IF levels and adjusting the attenuators. The hardware ought to be able to accomplish that in milliseconds.
6. Correlator set-up. It may be hard to decide here what overheads are intrinsic to the hardware and what to software since there is a great deal of software that is directly associated with the correlator. Changing between two different frequency domain modes has a 1.5 second latency but it may be possible to switch between frequency domain and time domain more rapidly than that.
7. Subreflector drives. These are specified to be fast enough to keep up with the required changes in focus as the elevation changes even in a slew. There is an overhead when doing focus measurements because as a minimum we have to move about +/– half a wavelength to find the optimum Z-focus. (Check details here.)
8. Nutator. The spec calls for this to have an 80% duty cycle when chopping at 10 Hz but it would be a bit unreasonable to include that in this list of hardware overheads. I don't think there is any other overhead in the specs but it might be wise to allow say 5 seconds to get it set up and running.
9. WVR. No significant overheads once operational. Perhaps occasional (weekly) checks with the calibration loads?

So the next step is to add these up for the various benchmark Observing Blocks already outlined and get a figure for the minimum overhead fixed by the hardware in each case. The software overheads associated with each of the actions can then be estimated (and eventually measured). In addition to the actions listed for the hardware, a number of purely software items clearly need to be accounted for. I am not the person to cover this but I suppose they should include the interpretation of the Observing Block, assigning of resources, and cleaning up at the end of the process before the system can start the next Block and at lower level the starting and stopping of a scan and subscan, etc.

Richard Hills 6th Oct 2009