VLA COMPUTER MEMORANDUM #178

COMPUTING REQUIREMENTS FOR CALIBRATION OF FUTURE VLA DATA

January 1988

Rick Perley

I. INTRODUCTION

The Dec-10, which currently is used to calibrate virtually all VLA data, is scheduled to be 'retired' at a time no earlier than December, 1988. This computer performs a large number of essential tasks for the VLA, of which data calibration takes, by far, the largest amount of CPU time. In the future, all data calibration will be done within AIPS, using a set of tasks known as the 'AIPS Calibration Package". At the same time, the new MODCOMPs will be on line, with the potential for greatly increased data flow.

These changes will have major effects on our computing requirements, and it is obviously of great importance to attempt an estimation of what these will be. To do this, I have set up a 'RUNFILE' in AIPS, on both our VAX 11/780 and our CONVEX C-1, which fully edits and calibrates a 'typical' database, using 'standard' techniques. To allow detailed comparisons, I have also constructed a Dec-10 'MIC' file which performs essentially identical operations. In setting up these packages, I have attempted to be realistic at every step, mimicking the effort expended by an *experienced* user.

A primary goal was to measure the elapsed time ('wall-clock time') required to execute the package, on different machines, under different loading factors (i.e., running 1, 2, 3, and 4 independent jobs in parallel). To allow easier interpretation, in almost all cases the machines were arranged to have no other jobs running at the times of the various tests. Exceptions to this will be noted in the detailed descriptions. It is also of interest to record both the CPU and elapsed time usage for each task, and this has been done in most cases. CPU times for the DEC-10 are not well-defined (being strongly dependent on load factors), so only elapsed times were recorded.

The results of these benchmarking tests are described in Sections II through V. Section VI is a short discourse on the new AIPS task TVFLG. In the last section, I attempt an estimation of our future computing needs to support data reduction by taking the current calibration throughput of the Dec-10 and adjusting it by various factors (such as increased data flow by the new MODCOMP system, and external calibration by users), to produce an estimate based in 'equivalent Dec-10 units'. This is then converted into CONVEX or VAX units by using the ratios determined in Sections II to V.

It must be pointed out that the use of these procedures has reduced to near zero the 'human time' required in calibration – that time required to think about the inputs, read the instructions, make and correct mistakes, etc. Most importantly, the package 'knows' where the bad data are, so that the (usually) considerable time taken to find bad data (by whatever means) has been reduced to zero. To reach a 'totally realistic' elapsed time, one must add in the time expended by an individual in executing the various steps. This is obviously highly dependent upon user experience and data quality. I have made no attempt to quantify these times. Clearly, these times will greatly affect the required disk space and, through system inefficiencies, will influence to a lesser degree, the required CPU.

II. THE DATABASE AND RUNFILE

I selected an 'A'-configuration run of 4-hours duration at 6-cm wavelength for the tests. The averaging time was 10 seconds, resulting in about 288,000 visibilities in each of the 2 IFs. (Here, we use the AIPS convention: IF 1 corresponds to the 'A/C' IFPairs, IF 2 to 'B/D'). This combination represents a sort of 'average' continuum database. This database was chosen mainly for

its convenience, not for its quality. As it turned out, the data are of exceptional quality, requiring almost no flagging.

The calibration paths cannot be made exactly identical, since the Dec-10 route requires two tasks, EXPVIS and UVLOD, which have no parallel in the AIPS system. Conversely, the AIPS calibration package requires SPLIT, and (temporarily) DBCON and INDXR, which are not required in the Dec-10 route. All these extra tasks were benchmarked as well. After loading the data on the appropriate computer, and running DBCON and INDXR for the AIPS calibration path, the RUNFILE/MICfile performed the following operations (all processing was on both IFs):

1. A scan and summary listing of the database.
2. A Matrix listing of both IFs, with ampscalar scan averaging and rms. Calibration was NOT applied (a significant difference for all machines).
3. 'Column' listings of small amounts of data, required to find the few bad data points.
4. Flagging the bad data. I flagged both a single record for all antennas, and an entire scan for a single antenna. (Only the former was required. I ran the latter, just to see if there was any significant difference in execution time.)
5. Setting the flux density of 3C286.
6. Running CALIB on 3C286, both IFs.
7. Listing the solutions of the above.
8. Applying these solutions with CLCAL to all data.
9. Running LISTR to list the amplitudes of the phase calibrators. This step must be run twice, once for each IF.
10. Repeating steps 5 through 8, with the two phase calibrators instead of 3C286.
11. Running PCAL to determine the antenna polarizations.
12. Running LISTR, twice, to show the RL and LR correlations, and to determine the R-L phase correction.
13. Running CLCOR, to apply the above correction.
14. Repeating 12, to check that the correction is correct.
15. Running LISTR, calibration applied, amplitude and rms output.

Polarization calibration procedures on the Dec-10 differ slightly from the AIPS route, so the MIC file differed slightly in the final steps. However, the same operations were carried out.

I expect that many will question the necessity of step (15), since a similar (and cheaper) LISTR was run in step (2). However, cautious observers will often execute this step as a final check. I would expect that even when the wondrous TVFLG is widely used, observers will still execute a final matrix listing. For the purposes of comparing throughput, the actual path chosen is less important than ensuring the path is the same on all machines. I will comment on the impact of omitting unnecessary steps, and on anticipated code improvements in the next section.

In the RUNFILE, I have inserted numerous 'INP taskname' commands, to simulate a user's actual effort. On the Dec-10 and CONVEX, these insertions have negligible effect on the run times. However, on the VAX, a very different picture emerged, as may be expected by anyone who has tried to type 'INP' on a loaded VAX. Summing the total elapsed times of all tasks, and comparing to the wall-clock time, as recorded in the logfile, of the entire procedure gives an estimate of the loading caused by typing 'INP xxx'. On the VAX, the minimum difference between these times was 12 minutes. Monitoring the CPU usage ('MON PROC/TOPCPU') shows that 30% of the VAX CPU is required to write the inputs onto the terminal screen! (when the machine is running no other jobs). Those who disbelieve this figure are invited to prove it wrong!

III. SUMMARY OF RESULTS

Perhaps the most important statistic is the elapsed time for fully calibrating ones data – from the time the MODCOMP tape is mounted to the time one is ready to image the data. These times

are shown, in seconds, in Table 1. Displayed are the total throughput times when running one job on an otherwise unloaded machine.

In this table, the first column displays the appropriate task of operation. The second and third columns give the total elapsed time to execute these operations, when following the 'Dec-10' calibration route. The two different columns indicate the final destination of the Dec-10 calibrated data. Note that the times differ only for UVLOD and DBCON. X's mean that the listed task is inappropriate for the route. Question marks following some of the CONVEX entries indicate that I was unable to obtain an idle machine when the task was run, so the time listed is my best estimate. I expect the error to be less than 10% in all cases. The fourth and fifth columns list the elapsed times following the 'AIPS' calibration route. The last column contains a comment on the operation performed.

The block of four tasks near the top of the Table list the operations necessary to load the data onto the various machines. The Subtotal listed is the total of these operations. The next row, labeled 'Calibration', lists the elapsed times to perform the full calibration. The detailed breakdown of the tasks within this is given in the Table 4. The next block of four tasks lists the required times for transferring the data to a form ready for imaging, with the total given in the next-to-last line. The final line gives the grand total elapsed times.

This chart contains some interesting values. In terms of total throughput, the AIPS calibration package, on a CONVEX, is about twice as fast as following the Dec-10 Calibration route. However, the AIPS calibration package on the VAX is 50% slower than the Dec-10 route. Note, however, that the line listed 'Calibration' shows distinctly different ratios. Here, the CONVEX outperforms the Dec-10 by a factor less than 1.5, while the VAX is now slower than the Dec-10 by nearly a factor of three. These differences are due mainly to the extra operations of EXPVIS, UVLOD and DBCON, required in the Dec-10 calibration path.

Table 1. IDLE MACHINE RUN TIMES FOR CALIBRATION (seconds)

TASK	STD. CAL.		AIPS CAL.		Comment
	VAX	CONVEX	VAX	CONVEX	
FILLER	2340	2340	X	X	Fill onto Dec-10
FILLR	X	X	3173	1000(?)	Fill into AIPS
DBCON	X	X	637	235	Join the two databases
INDXR	X	X	136	20	Create Index file
SUBTOTAL	2340	2340	3946	1255	Total for loading data
CALIBRATION	3300	3300	9247	2445	Total for Calibration
EXPVIS	2100	2100	X	X	Export data from Dec-10
UVLOD	630	200?	X	X	Load Dec-10 data on AIPS
DBCON	670	175?	X	X	Join IFs into 1 database
SPLIT	X	X	3007	750?	Separate Multi-source dbase
SUBTOTAL	3400	2500	3007	750	Total for Data exporting
GRAND TOTAL	9040	8115	13200	4450	

Considerable improvements in throughput may accrue from changes in both procedure and code. If we delete the final LISTR from the processing path, the total runtime decreases by approximately 20% on the CONVEX, and 30% on the VAX. In the near future, the AIPS tasks

DBCON and INDXR will not be required (as the new MODCOMP format will usually hold an entire database on one tape). Deleting both these tasks, and the final LISTR, reduces the run times by 25% for the CONVEX, 35% for the VAX. Finally, it is entirely possible that speedups of, say, 30% are possible in the code. Combining these factors, we might predict a final runtime half of the values listed in the bottom line for the AIPS CAL. route. In terms of the currently available Dec-10 route, an optimistic estimate is that the throughput of the CONVEX will be four times, and for VAX, slightly better than, the current throughput. Recall that these figures are valid for single jobs running on an otherwise unloaded machine.

IV. THE IMPACT OF MACHINE LOADING

The values given in the last section would be all that are needed if we knew that only one job was running at a time, or that the machine responses were perfectly linear with load factors. Neither of these conditions are expected to occur, so numerous tests were performed to measure the machine reaction to loading. As stated in the Introduction, this was measured by running the procedures in parallel. It is difficult or impossible to run the tape I/O jobs in parallel, so only the calibration portion was tested this way.

In Table 2 are the total elapsed run times, in minutes, for the procedure described in section II. Column 1 gives the load factor, defined as the number of procedures running in parallel. The remaining columns give the elapsed times in minutes. For the VAX and CONVEX, no other jobs were running. For the Dec-10, only the entry for loadfactor = 4 has been notably affected by other users. Perhaps 10 or 20 minutes need to be subtracted. The entries do not tally exactly with the entries in Table 1, since the AIPS tasks DBCON and INDXR are added. Removing these reduces the VAX and CONVEX entries by about 10%.

Table 2. TOTAL ELAPSED TIMES (minutes)

#	Dec-10	VAX	CONVEX
1	55	167	45
2	81	320	63
3	110	510	93*
4	165	[700]	[125]

The entries in square brackets have been predicted, using values under smaller load factors. The starred Convex entry is partially predicted, since the Runfile did not run to completion, requiring me to estimate the additional time. This was a small quantity, so an error in this estimate is of little consequence.

It will be noted that the VAX run times are nearly linear, indicating that the jobs which dominate the elapsed time (LISTR, CALIB, and PCAL) each saturate the machine. This was confirmed by monitoring the system CPU usage – each of these tasks used typically 80% or more of the CPU. Both the CONVEX and DEC-10 responses are distinctly non-linear, with saturation for the Dec-10 not occuring until 4 jobs are running concurrently.

In table 3, I present the run times normalized by the Dec-10 run time. The final line is somewhat questionable, as it utilized predicted times for the Convex and VAX, as well as a curiously long run time for the Dec-10, (which may be explained by the presence of some other users on the system when the test was run.)

Table 3. RUN TIMES NORMALIZED BY DEC-10 RUN TIME

#	VAX	CONVEX
1	3.0	0.80
2	4.0	0.78
3	4.6	0.85
4	[4.2]	[0.76]

The actual entries in this table are less important than their trends with load factors. I believe the central points here are: (1). The CONVEX and DEC-10 react very similarly under loading, so that runtime ratios derived in Section III can be used for all practical load factors. (2). The runtime ratios between the VAX and DEC-10 are strongly affected by loading, such that under realistic load factors (greater than 3), the rough equivalency of these machines under a single job will become greater than a 2:1 ratio against the VAX.

V. DETAILED BREAKDOWN OF RUN TIMES

Different tasks run at different rates on the machines, and the ratios between run times are different between machines. This is presumably due to the differing mix of I/O rates, CPU, and how much of each is required by each task. To assist in judging what tasks need to be optimised, I present in Table 4 the elapsed times, per task and per loading factor. In addition, for the AIPS tasks, the CPU times are also recorded. In all cases, the format is

CPU time/Elapsed time.

This breakdown is not easily gotten for the Dec-10, since the 'CPU-time' on this machine is a strong function of the loading. Rather than 'compare apples with oranges', I have opted for listing the elapsed run times for a single job on an empty machine.

Clearly, the tasks which should be looked at for optimization are LISTR (and/or TVFLG, see the next section), CALIB, PCAL, and perhaps CLCAL. After this, it clearly would be useful to attempt improvement of FILLR and SPLIT (see Table 1 for comparison).

VI. THE AIPS TASK TVFLG

It will quickly be noted that the task LISTR dominates the run times. It is certainly wasteful to produce such extensive listing of data when, in general, so little requires purging. Further, having the listings is only the beginning, as one must run column listings to actually find the times needing flagging. The user time required in these operations can be enormous.

The new task TVFLG will probably revolutionize this aspect of data processing. I believe that it will largely replace LISTR as the chief instrument for identification and purging of corrupted data. One must note, however, that although it is *easier* to use that LISTRs, it will not be any *cheaper*. To estimate the impact of TVFLG, I ran this task on the data on both the CONVEX and VAX. I did virtually no flagging, so the times reported are for making the BT map. The CPU/Elapsed times (multiplied by two, in order to account for both RR and LL polarizations, since TVFLG does only one at a time) are 620/750 for the CONVEX, and 2344/2800 for the VAX. Note that these are very comparable to the initial LISTR, with RMS's. Use of TVFLG will also eliminate the need for column listings, but these take very little machine time, so their elimination will not affect the basic conclusions. I make here the obvious point that widespread use of TVFLG will require many more TVs.

So, for the purposes of estimation, we may replace the first LISTR in the procedure with TVFLG, with essentially no adjustment in runtime or CPU time. (Recall that I am not attempting to estimate the 'human time' factor, which almost certainly will be improved with TVFLG).

Table 4. CPU TIMES AND RUN TIMES FOR VARIOUS TASKS

AIPS TASK	DEC-10	VAX		CONVEX			Comments
		1 job	2 jobs	1 job	2 jobs	3 jobs	
DBCON	-	523/637	535/1257	159/235	152/330	154/470	Unneeded in Dec-10
INDXR	-	41/136	44/251	19/20	19/40	20/57	Unneeded in Dec-10
LISTR	3	3/9	9/16	1/5	1/5	1/6	Scan, summary listings
LISTR	815	1087/1219	1096/2365	348/494	308/620	312/948	Ampsc. amp&rms, No cal.
LISTR	14	82/132	88/233	28/44	28/60	28/84	Four scan-length column listings
UVFLG	18	2/7	2/11	1/2	1/2	1/3	Flagging part of one scan, all ant.
UVFLG	4	1/4	1/7	1/2	1/2	1/3	Flags all of one scan, one antenna
SETJY	2	3/10	3/12	1/1	1/1	1/1	Set flux of 3C286
CALIB	20	46/92	48/149	16/28	17/45	17/58	Solution of 3C286
LISTR	10	14/45	15/64	2/17	2/9	2/13	List solutions, Amp and phase
CLCAL	12	94/400	96/496	41/70	41/93	44/147	Apply above solutions, boxcar
LISTR	160	316/392	328/708	90/108	92/204	?	List calibrator amplitudes
SETJY	4	6/20	6/21	2/2	1/1	1/1	Set fluxes for 2 sources
CALIB	230	360/460	367/784	118/144	123/250	123/371	Solve for gains on all cals.
LISTR	18	55/116	57/148	17/32	14/36	14/45	List solutions for all cals.
CLCAL	23	87/375	90/468	40/65	48/109	51/162	Apply cals, 2-point interpolation
PCAL	154	496/859	503/1226	135/200	134/283	137/412	Solve for antenna polarization
LISTR	26	80/176	error	24/32	error	error	List R-L phases
CLCOR	20	28/112	30/127	12/14	12/25	13/37	Rotate R-L phases
LISTR	27	78/173	89/383	24/30	?	error	Check R-L phases
LISTR	1709	3586/3949	3797/9346	792/896	782/1578	error	Amps. avg. &rms, Cal. Applied

VII. AN ATTEMPT TO PREDICT FUTURE COMPUTATION NEEDS

I have adopted the following method to estimate future CPU needs: Convert our current calibration effort into 'equivalent Dec-10 units'. Then adjust by known or predicted factors to allow for expected changes in data flow, changes in calibration path and techniques, changes in code, etc. Finally convert to the hypothetical future calibration computer by the throughput ratios. Note that this method automatically accounts for the 'inefficiency' factor of inexperienced users, since we are using current usage of the Dec-10 as our benchmark.

The various factors I have considered are probably incomplete, and the values derived are certainly not beyond question. I list below these factors, and the considerations which figured in their estimation.

1. Usage of the Dec-10. To estimate that fraction of the Dec-10's CPU which currently goes to calibration, I monitored the total Dec-10 CPU usage over a typical 5-day period. Approximately 60% of the available Dec-10 CPU time available was used. My estimate of the fraction of this which supports Dec-10 Calibration is 90% (others agree). Thus, we can state that close to 50% of the available CPU is used to calibrate data. This is a large fraction, and agrees well with all users' impressions that the Dec-10 is essentially running at full capacity most of the time. It is essentially impossible to recover all, or even most, of the unused CPU time, as this would require rigid scheduling of when calibration could be done, as well as probably expanding the remote calibration group. I would expect the loading of any future calibration computer to be very similar to that experienced on the Dec-10 – fully saturated during daytimes, mostly saturated in evenings, fairly empty at night. As pressure mounts, more calibration will naturally be done at night, but I doubt we could ever reach 100% usage, all the time. Probably half the unused CPU time could be recovered, leading to a factor of 0.75.

2. Spectral Line Data (Current Data Flow). Most calibration of spectral line data is already done on the Dec-10. The important exceptions are the production of BT images, to allow editing, and the actual application of the determined gains. These tasks are done by the pipeline. The cost of producing a BT image on the Dec-10 should be similar to producing a LISTER, with rms. This process takes about 1/6 of the total run time, and since 1/3 of all observing time is in spectral line, the added cost of this operation is very small, perhaps 5% of the current load. Much more important is the effect of applying gains. Both antenna gains and passband gains must be applied. Were both of these to be done on the Dec-10, the programs SPECTR and EXPVIS would need to be run repeatedly – a horrendously expensive route. I will instead presume the existence of a spectral-line EXPVIS, whose cost is proportional to data volume. Noting we are currently taking equal volumes of spectral line data as continuum, that applying the gains takes about 1/3 of the processing time, and doubling this factor to allow for passband calibration, leads to a factor of about 1.7. Because of other uncertainties in procedure (such as the probable future use of more complicated flagging routines to handle low-frequency spectral line), I feel a realistic estimate of this factor is 2.

3. The Effect of the New Modcomps. The effect of turning on the new MODCOMPs will clearly have a major effect on both data volume and CPU requirements. In spectral line, the volume of data taken in this mode will greatly increase, potentially by 20-fold. Most programs will not utilize this explosive factor, however. Basic calibration (done on channel 0) will mainly be affected by the larger number of baselines. However, this portion of calibration takes a small fraction of the total CPU. Bandpass calibration will also be more expensive, since most of the data increase will be due to a larger number of channels. Bandpass calibration is about as expensive as antenna gain calibration so I expect the increased CPU requirements for this process also to be small. However, the effect of the data volume increase on CPU times to apply the derived gains and flags should be important. And, if editing on the basis of BTF cubes becomes common (which

is quite likely at 327 MHz), there will be great increases in required CPU to support this too. We should also consider the possible non-linear factor which makes calibration more expensive with increased database size (due to I/O considerations). My best guess is that calibration of future spectral line data will require triple the current capacity.

The main effect of the new system on continuum observing will be to encourage a small fraction of programs to observe in polarization spectral line mode with a shorter time integration. The calibration of these types of programs will be very expensive, but fortunately, the total impact will be diluted by the small number of programs requiring (or requesting) this mode. I guess a doubling of current capacity will be required to handle increased continuum. Overall, I will use 3 as the factor required to handle the increased data flow from the new Modcomps.

4. Code and Procedural Improvements As stated in Section III, improvements in both procedure and code are likely to increase throughput by a factor of 2. Perhaps greater factors are possible in the future, especially with regard to spectral line processing.

5. At-Home Calibration I don't believe that there will be any great savings in the fact that most observers will be able to calibrate their data 'at home'. Many of these users don't have the resources, especially the disk space, to calibrate data. Many observers will opt to use our systems, where resident experts are around to help them out of their difficulties. (Travel costs are cheap compared to machine costs). And, I don't believe there will be any significant reduction in remote calibration. Any time one offers a free service, with the results essentially guaranteed, there will be no lack of takers. My best estimate for this savings is 0.75. Optimists might prefer 0.5

I summarize these arguments in Table 5. Factor is defined as the rate by which data calibration throughput will be retarded by the listed consideration. The four factors are multiplied to arrive at the predicted product factor. This is then converted into needed CONVEXes and VAXes, using the ratios determined in Sections III and IV.

Table 5. SUMMARY OF DATA THROUGHPUT FACTORS AND CALIBRATION NEEDS

Consideration	Factor
Dec-10 Usage	0.75
Spectral Line	2
New Modcomps	3
Code/Procedure Improvements	0.5
At-Home Calibration	0.75
Product Factor	2
Conversion to CONVEX	0.5
Needed CONVEXes	1
Conversion to VAX	3
Needed VAXes	6

The 'bottom line' is that one CONVEX C-1 will be required to handle all our future calibration needs.