EVLA MEMO 24

COMPUTING FOR EVLA CALIBRATION AND IMAGING

T.J. CORNWELL

Abstract. The EVLA will bring large improvements in all areas of scientific performance. To realize many of these improvements will require processing of the observed data at a level considerably more demanding than that currently required for the VLA. The important questions considered here are (a) whether the computing is feasible? (b) what is the best computing model to use? and (c) how much would the necessary compute power cost when purchased for the EVLA?

In developing answers to these questions we have used two different approaches: first, general scaling arguments based on Moore's law, and, second, detailed scaling arguments based upon the knowledge of the main sources of computing costs in processing various types of observations. We find that these two estimates agree reasonably well, and indicate that the computing load for the EVLA is relatively modest in scale (for 2009).

We also discuss the software development needed for EVLA calibration and imaging. We find that while the current AIPS and *AIPS++* packages can support the simpler observational modes of the EVLA, full exploitation of all the capabilities will require some software development, but none beyond that already occurring within the context of *AIPS++*.

1. Is the computing feasible?

In this section, we use general scaling arguments to investigate the feasibility of the computed required for the EVLA. The feasibility is determined by the playoff between a number of scientific factors and by the way that computing itself evolves between now and the commissioning date. First we consider the scientific factors. The key scientific factors are:

- Overall data rate and volume
- Typical data quality
- Acceptable turnaround to a scientific result
- Level and type of computing required for a scientific result
- Relative importance of one-off batch processing versus interactive processing.
- The spectrum of scientific observations scheduled on the array
- The necessity to allow growth in computing requirements

We examine all of these factors in turn.

Data rate and volume: After noting that the peak processing rate could potentially be very large, the EVLA scientific specifications (Benson and Owen, EVLA Memo 15, 1999) argue for a compromise of 20 - 25 MB/s.

Date: 2001 January 12.
I thank Brian Glendenning and Athol Kemball for useful discussions.

Previous studies have shown that between 100 and 10000 floating point operations per floating point value are needed to image radio astronomy data. Thus to keep up with the incoming data, the range of *peak* data processing is 0.5 Gflop - 50 Gflop. The data volume for an 8 hour observation would be roughly 720GB. A day's observation at this rate would be about 2TB, equivalent to a few years observing with the current VLA!

Data quality: In the early days of the VLA, data reduction was complicated by the relatively poor data quality which necessitated close inspection and editing of the observations. As the system was debugged, the quality improved considerably with a concomitant decrease in the level of inspection and editing required. The EVLA will have the same problem, of course, for a few years, but even after that radio frequency interference may limit data quality. In any event, tools for the assessing the quality of large data sets will be important.

Acceptable turnaround: The ratio of observing time to acceptable turnaround time magnifies the peak processing rate proportionately. Thus *for example*, if we wish to reduce an 8 hour observation in minutes, the peak processing rate is magnified by about a factor of 100. This factor is tied to the need for experimentation in the data reduction: if many of the largest experiments must be processed a number of different ways then the acceptable turnaround time will be small. Using the factor of 100, the peak processing rate would be about 5 Tflop.

Peak data rate	25 MB/s
Data for Peak 8-hr observation	700GB
flops per float	100 - 10000
Peak compute rate	5Tflop
Average/Peak computing load	0.1
Average compute rate	0.5Tflop
Turnaround for 8-hr peak observation	40 minutes
Average/Peak data volume	0.1
Data for Average 8-hr observation	70GB
Data for Average 1-yr	80TB

Table I: Typical and peak data and computing rates for the EVLA

Level and type of processing: The range in processing required for various scientific experiments varies tremendously. A simple example would be a long detection experiment at greater than arcsecond resolution. This would require straightforward calibration and construction of a dirty image only (assuming no other sources in the field of view). More complicated examples abound, ranging from multi-pointing mosaics to very high dynamic range imaging of an entire primary beam, where time consuming deconvolution and self-calibration would often be necessary.

Batch versus interactive processing: We expect that both batch and interactive processing will be required. Some of the simplest observations are amenable to batch processing, but we know that in the past many of the most spectacular scientific results have relied upon considerable hands-on processing by expert users. The EVLA will not change this mixture.

Spectrum of scheduled observations: The peak throughput is set by turn-around but the average is set by the mixture of science scheduled on the array. A reasonable factor here would be 0.1 or less *i.e.* the average project has a tenth or less of the computing requirements of the most difficult projects. We will return to discussion of this factor below.

Growth in computing requirements: Based upon our past experience, we can say with certainty that the level and type of processing will continue to evolve, perhaps roughly as Moore's law. Thus any model must scale.

Thus we have the following interesting conclusions (see Table I).

We know the wonders of Moore's Law [1] but there are related cost-scaling laws that also must be considered in answering these questions. We have to know the relative cost of processing, network bandwidth (both local and wide area), disk storage, and software, all of which must be predicted out about 8-9 years to the expected commissioning date of the EVLA. For many of these scaling laws, we have used an excellent comprehensive overview[2] compiled by Jim Gray and Prasant Shenoy of Microsoft Research.

A few selected rules of thumb are:

1 Moore's law: Things get 4x better every 3 years
2 You need an extra bit of addressing every 18 months
3 Storage capacities increase 100x per decade
4 Storage device throughput increases 10x per decade
7 NearlineTape:OnlineDisk:RAM storage cost ratios are approximately 1:3:300
8 In ten years RAM will cost what disk costs today
9 A person can administer $1M of disk storage
14 Gilder's law: Deployed bandwidth triples every year
15 Link bandwidth increases 4x every 3 years

We can now draw some conclusions about the feasibility of EVLA computing, assuming that we deploy a computing solution in 2009:

- 2000 - 2009 is three 3-year cycles, so for Moore's Law scaling (RAM, CPU performance, and network bandwidth), we can expect a factor of 64 improvement over 2000.
- The growth of telescope data rates (from VLA to EVLA) is sub-Moore's law, and so on this very basic level, the computing must be feasible.
- Storage capacity growth is roughly Moore's law or better, whereas storage device throughput growth is significantly less. Thus storing the data will not be more of a problem than it is now but minimizing the number of I/Os per flop will become more important.
- Optimum algorithms will change as Moore's law enables simplifications and short cuts (*e.g.* eventually many images will fit into the available fast-access memory.
- The current cost of storing a TB in disks is about $50K - $100K, so the cost of storing 100 TB (a year's observing) in 2009 will be roughly the same.

[1] One important question is how long Moore's Law will continue. Moore himself has spoken recently (2000) about this topic. He expects the law to hold for about another 10 years.

[2] http://research.microsoft.com/~gray/papers/MS_TR_99_100_Rules_of_Thumb_in_Data_Engineering.pdf

- Putting the same data onto a tape robot may save a factor of three cost, but that forecast is probably less certain than the others.
- To get 0.5Tflop in 2009, we would have to spend an amount that in 2000 would buy 500Gflop/64 = 8Gflop, roughly \$20K - \$100K, depending on how it is done.
- The total deployed bandwidth will be 3^8 times $\sim$ 6000 times greater, which is considerably super-Moore's law. Thus moving data will be much cheaper than today, but the transfer time for a typical VLA observation (in seconds) will be roughly as it is now and will therefore be possible in real time.
- All these arguments are based on deployment in 2009. Hence any purchases of computing capabilities prior to 2009 should be as circumscribed as possible.

So, in summary, from these rough arguments, we can see that EVLA computing will be feasible in 2009, and that the cost should be in the range of a few times \$100K. This means that the same computing would cost \$5M - \$10M, if deployed today, which is roughly comparable to the cost of the correlator. In this sense, the EVLA will be well-balanced.

Finally, in concluding this section, we should note that some of these improvements may be easier to come by than others. We can reply on Moore's law continuing for computers, disk storage, and local area networks. However, this is not necessarily true for wide-area-networks. For example, since the EVLA is physically just as remote as the VLA is (!), the actual provision of high-bandwidth network links is highly dependent upon the exact course of development by major bandwidth providers in western New Mexico. Consequently, we will need to continue to be pro-active in pushing our interests in networking within New Mexico.

2. A more detailed examination of computing time

The above analysis is strongly reliant on the assertion that the number of floating point operations per float has a final range of 100 - 10000. If this assertion is incorrect then the processing required could be correspondingly higher. For this reason, in this section we investigate a number of typical observational scenarios with an eye to determining this number in practice.

To come up with an independent number, we will use a simple model of various imaging algorithms. Most of the time taken in imaging (including deconvolution) lies either in the gridding step or in the Fast Fourier Transform. The other steps such as the minor loop for CLEAN can be tuned to be less than or comparable to these steps. The costs are as follows:

Gridding: To a first approximation, the work involved in gridding is simply proportional to the number of visibility points to be gridded. Some savings could be realized by *e.g.* gridding as MIRIAD does onto a grid that is frequency variable, but we will ignore such optimizations for the moment. De-gridding costs are very close to those of gridding. The total time is therefore given by:

$$T^{grid} = N_{griddings}.N_{mega-vis}.t_{mega-grid} \tag{1}$$

where $N_{griddings}$ is the number of gridding or de-gridding steps, $N_{mega-vis}$ is the number of millions of visibilities, and $t_{nega-grid}$ is the time taken to grid one million visibilities.

FFT: In theory, the time involved in FFTs is expected to scale as $N \log(N)$ but we typically find in practice that it goes roughly linearly in the number of pixels [3]. We therefore have:

$$T^{FFT} = N_{FFTs}.N_{mega-pixel}.t_{mega-FFT} \tag{2}$$

where N_{FFTs} is the number of FFTs required, $N_{mega-pixel}$ is the image size in mega-pixels, and $t_{mega-FFT}$ is the time taken to perform a 1024 by 1024 FFT.

The times for various types of processing can be analyzed in terms of these times.

Single image deconvolution: Typically the work involved per minor cycle of a deconvolution is 2 FFT steps. This arises because the usual work required is to calculate residuals from an iterate using a double-size zero-padded FFT-based convolution. Adding time for the gridding of the dirty image, PSF, and calculation of residuals at the end, and accounting for the required padding, we have:

$$T^{sid} \sim 4.N_{mega-vis}.t_{mega-grid} + 8.N_{cycles}.N_{mega-pixel}.t_{mega-FFT} \tag{3}$$

Multiple image deconvolution: If multiple coupled images on different tangent planes or disjoint images are to be estimated simultaneously, one will have to use FFTs plus gridding and de-gridding at each minor cycle instead of full-size zero-padded FFT-based convolution. This means that multiple image deconvolution is much more gridding/de-gridding intensive than single image deconvolution.

$$T^{mid} \sim (4+2.N_{cycles}.N_{images}).N_{mega-vis}.t_{mega-grid}+2.N_{cycles}.N_{mega-pixel}.t_{mega-FFT} \tag{4}$$

Mosaicing: Only a fraction of the visibilities need to be gridded for each pointing. Also, the FFTs need not be as large as the whole field since the primary beam is limited. Finally, FFT-based convolutions can be used in each minor cycle to avoid repeated gridding and degridding. The time is therefore:

$$T^{mosaic} \sim 4.N_{mega-vis}.t_{mega-grid}+8.N_{cycles}.N_{pointings}.N_{pointing-mega-pixel}.t_{mega-FFT} \tag{5}$$

where $N_{pointings}$ is the number of pointings and $N_{pointing-mega-pixels}$ is the size of one pointing image. Since on average, the pointings will be critically sampled in two dimensions, the total time can be written:

$$T^{mosaic} \sim 4.N_{mega-vis}.t_{mega-grid} + 16.N_{cycles}.N_{mega-pixel}.t_{mega-FFT} \tag{6}$$

where $N_{mega-pixel}$ is the size of the total image. This is an asymptotic value for large numbers of pointings where the edge effects are unimportant.

In this analysis, the source structure affects the number of cycles. For simple sources, the number of cycles can be 10 - 30, whereas for complicated sources, it can range into the hundreds.

[3] This is presumably because data I/O dominates the transform. *AIPS++* recipe number 4 demonstrates this scaling law very nicely.

3. Some examples

We are now in a position to estimate the computing times for various EVLA projects. We can use values of $t_{mega-grid}$ and $t_{mega-FFT}$ for typical machines now, and use Moore's law to convert those to 2009 values. For a current 450 MHz Pentium III PC with an Ultra-Wide SCSI disk, $t_{mega-grid} \sim 60$ sec and $t_{mega-FFT} \sim 12$ sec.

We will consider a range of "big" observations, ranging from many pointing mosaics to full sensitivity continuum imaging of the L-band primary beam.

Large RRI mosaic of SGRA West (from Miller Goss): The frequency range would be 28.27 (H 61 alpha) to 40.63 GHz (54 alpha H) . This allows observation of 8 Hydrogen recombination lines in the range 28.27 to 40.63 GHz *i.e.* the full Ka band. The synthesizied beam is 0.4 to 0.6 arc second. There is enough brightness in Sgr A West to image at this resolution. This is a modern version of the Roberts and Goss result at 1.5 to 2 arc sec at H 92 alpha (Ap J suppl, vol 86, page 133 , 1993). The spacings of the recombination lines is about 2 Ghz in the middle of the band .

The mosaic must be 8 by 6 pointings with a spacing of 25 arc sec . The primary beam is in the range 60 to 90 arc sec or so. One phase references using Sgr A star - at the center of the mosaic. A possible observation lasts for 8 hours or 480 mins, of which about 300 min is observing - with the rest bandpass and phase calibration *etc.*.

The resolution of each spectra needs to be about 5 km/s or 0.58 MHz at the center of the band . One needs a velocity range of about plus and minus 300 km/s or about 70 Mhz for 128 channels.

HI cube of nearby galaxy (from Rupen, EVLA memo 8): The frequency resolution should be about 6 kHz, and the full bandwidth 7 MHz. The field of view is about 10 arcmin, and the resolution 1.5 arcsec.

Noise-limited image of entire L-band primary beam: All 4 polarizations must be used to reach the theoretical noise level. All sources out to the second sidelobe of the primary beam must be included.

Some commentary at this point is worthwhile. Clearly the range of processing times is huge: the reduction of the entire L-band primary beam takes much longer than any of the other types of processing. The time is driven by the number of images than are needed to represent the non-isoplanatic imaging adequately over the primary beam. In this regime, it may be worth reconsidering the algorithm used for wide field imaging. As Cornwell and Perley (A&A, 261, 353, 1992) note, there are many viable algorithms to choose from, and if the relative cost of various computing resources changes (as it must do over time), the optimum algorithm may well change. Cornwell and Perley demonstrated two algorithms: a faceted transform and a three-dimensional transform. The former wins currently because of the limited memory sizes available. However, if the memory allows the three-dimensional transform to be used then the table shows that the computing time can be cut very substantially. The rules of thumb above tell us that this is likely to be true, so we should assume that the lower number is appropriate.

Another factor to be considered is the important of self-calibration. For the spectral line cases, the incremental cost is relatively small since the self-calibration will often be done on the pseudo-continuum data. For the high dynamic range

Observation	# pol	FOV "	Cellsize"	# pointings	# facets	# pixels	BW (MHz)	Freq res (MHz)	# vis chan	#image chan	#I.F.s	T obs (hr)	T int (sec)	# vis / int
L-band full primary beam (2D)	4	7200	0.3	1	256	24000	500	1.00	500	1	1	12	3	702000
L-band full primary beam (3D)	4	7200	0.3	1	1	24000	500	1.00	500	16	1	12	3	702000
RRI Mosaic of SGRA West	2	200	0.2	64	1	1000	70	0.5468	128	128	8	8	10	718848
HI of nearby galaxy	2	600	0.5	1	1	1200	7	0.006	1166	1024	1	24	10	818532

Observation	Data rate (MB/s)	Total data (GB)	Mpixel	Mvis	Minor cycles	single (d)	multiple (d)	mosaic (d)	Time (d)	# processors	TB/year
L-band full primary beam (2D)	1.87	80.87	576	10108.80	10	28.50	35972.08	40.88	35972.08	71944.16	59.04
L-band full primary beam (3D)	1.87	80.87	9216	10108.80	10	130.48	194.08	232.88	130.48	260.96	59.04
RRI Mosaic of SGRA West	0.58	16.56	128	2070.28	100	19.97	296.85	34.20	34.20	102.59	18.14
HI of nearby galaxy	0.65	56.58	1679.04	7072.12	10	38.30	122.53	56.96	38.30	38.30	20.65

Figure 1. Table II: Quantitative analysis of EVLA computing

examples, the cost is basically a multiplicative factor between 5 and 10, as the entire imaging must be repeated after each self-calibration. Difference imaging techniques may reduce this number to 3 - 5.

In summary, we see that for the typical (worst) case in our examples, the processing can be done in real time using a parallel processor of 100 (5 * 260) (year 2000) processors. Moore's law from 2000 to 2009 gives a factor of 64, so we would need roughly the equivalent of about 10 (20) (year 2009) processors. Depending on the achieved parallelization speedup (see Appendix A), this could require between 10 (20) and 100 (400) processors. Specifying for the average case, we find that the total hardware cost would be in the range of about $50K - $200K.

4. The computing model

We have seen from the arguments above that a moderately parallel computer will be required to reduce data from the most demanding EVLA observations. For many lesser observations, a more standard desktop (but of the year 2009!) will probably suffice, and the networks and local disk space will be available as required to permit observers to process some data at home, if they so desire. By the same argument, many (but not all) projects could be processed only when the data are demanded from the archive. This would allow processing schemes to be continuously updated and improved.

Hence the computing model has the following aspects:

- A moderately parallel machine ($50K - $100K) will be required for the high end projects.
- Standard (2009-issue, probably moderately parallel) workstations will continue to be able to process many projects.
- A central archive costing about $50K - $100K per year will be required.
- Data access will be over the Internet, with transmission times for entire data sets being typically a few hours.
- Calibrate-and-image-on-demand is possible for many projects, and perhaps for all.
- Processing capabilities should be available and accessible remotely.

NRAO currently has two packages that can perform many of the necessary algorithms for EVLA processing: AIPS and *AIPS++*. Over the last twenty years, AIPS has provided the processing required for the VLA. *AIPS++* has been designed and developed both for that class of processing, and for the high data volume and pipelined processing necessary for new telescopes such as the EVLA and ALMA.

Our overall scheme for producing the software required for EVLA data reduction can be summarized in the following table:

	Interactive	Pipelined data reduction
High-end reduction	*AIPS++*	*AIPS++*
Low-end reduction	AIPS, *AIPS++*	*AIPS++*

Table III: Software strategy for the EVLA

where we have defined:

low-end reduction: single-field or small mosaic imaging with small data volumes (10's MB up to several GB) on a single CPU.

high-end reduction: multi-field imaging with large data volumes (up to 100GB) using sophisticated visualization, editing, calibration and imaging tools, running on parallel computers.

interactive: the observer is responsible for overseeing and executing the data reduction.

pipelined reduction: automated data reduction driven directly from the observing schedule.

Thus much of the processing of single field observations could be done with the current AIPS and *AIPS++* packages, *without any addition modifications*. We doubt that AIPS will still be in *widespread* use in 2009, but it is possible. This provides a guaranteed baseline of performance that would accommodate many of the observing modes of the EVLA. As the capabilities of single CPUs improve over time, this guaranteed baseline will include more of the simpler observations of the EVLA but even so not all of the full range of capabilities of the EVLA will be accessible. Going beyond the bottom left corner of this table brings new demands on the software. These we discuss in the next section.

5. Software development

In this section, we discuss the software development needed to fulfil all the capabilities of the EVLA.

5.1. **High data volumes.** As described above, the data volumes for the VLA will increase by several orders of magnitude. Fortunately, the cost of data storage will drop to more or less compensate. However, the speed of access will not rise proportionately. This reinforces the importance of I/O in the overall computing cost estimates.

The key software requirements to deal with high data volumes are:

Decoupling logical from physical descriptions: As much as possible, the interfaces for data access should not assume any particular physical storage. Instead the interface should be based on a logical view of the data.

Efficient storage: Spending flops to minimize I/O will be more advantageous. Thus compression of data by a variety of algorithms (*e.g.* Run-Length-Encoding, optimal compression) will be very worthwhile. The optimum algorithm will be different for different data elements in different contexts.

Efficient access tuned to the range of access patterns: For images, one will want to access along all axes, and for visibility data, one will want to access the fundamental data along a number of random parameters: time, baseline, frequency, polarization. Rather than optimizing for just one access pattern, one should optimize for the expected range of access patterns.

In *AIPS++*, the Table system has been designed from the very beginning to support such requirements. The Table system is now very mature and stable, the bulk of the development having occurred early on in the project.

5.2. **More realistic calibration and imaging models.** Our understanding of how to best process synthesis telescope data has evolved over the years of operation of various telescopes. We have moved to more and more explicit recognition of various instrumental effects. We now understand the advantages of using encapsulated, modular, parameterized descriptions of various physical effects in the measurements performed by radio telescopes. We also understand the importance of models that can encompass both synthesis and single dish radio telescopes.

In *AIPS++*, the *MeasurementEquation* framework of classes has been designed from the beginning with such goals in mind. The key elements of this framework are:

- A specific, flexible and complete description of the measurement process for radio telescopes is built into the C++ classes.
- The model of the measurement process may be extended by plugging in C++ classes (*MeasurementComponents*) that describe specific calibration effects such as parameterized bandpasses or phase-screens.
- Generic algorithms for calibration and imaging are provided as part of the framework so that calibration and imaging can always be performed for any physical effect that can be described in the framework.
- Well-known algorithms, such as mosaicing, are available automatically from the framework. Extensions to more complex physical effects such as beam squint was relatively easy and automatic.
- Two complementary sky brightness descriptions are available: via images and via discrete components. The combination allows high dynamic range imaging in the presence of extended emission.

The MeasurementEquation formalism was prototyped in 1996 (Cornwell and Wieringa, 1997), and since then has been under revision, extension, and testing. A wide range of imaging modalities are now supported. Mosaicing and calibration have been the main focuses of development over the last year. More work is required in optimization, and parallelization, and in the streamlining of actual use of the package for calibration and imaging.

5.3. **Algorithmic flexibility.** As illustrated in the discussion of the processing of wide-field images from the VLA, the optimum algorithm is determined partly by the computer hardware on which it is to run. We saw how the advantage shifts from faceted to three-dimensional transforms as the typical fast memory size increases. Another example from the eighties was the tuning of CLEAN algorithms to the availability of the FPS AP-120B array processor. We expect other such shifts to occur continuously. This argues for a very flexible software environment in which substantial algorithms changes can be made with minimal software cost. While this is of course very difficult to ensure in all circumstances, well-engineered interfaces coupled with a high-level language such as C++ help considerably.

The MeasurementEquation framework in *AIPS++* supports such algorithmic flexibility by decoupling many effects. For example, the Fourier transform (faceted or three-dimensional) is independent of the deconvolution or calibration algorithms.

5.4. **General data formats.** To accommodate more complex data processing, one must use a more complete and general data format in which all the relevant information is stored. The data format must support the calibration and imaging models used, both for synthesis and single dish radio telescopes. The data format must allow storage of all the information needed for the calibration and imaging models. For example, for mosaicing, the antenna pointing position must be describable as a parameterized function of time. Another consideration is that the data format must easily accommodate the extraordinary data collecting flexibility of the WIDAR correlator in which the spectral setup may change completely from one scan to another.

The MeasurementSet data format in *AIPS++* has been through extensive use and testing, as well as one very extensively reviewed round of revision (see *AIPS++* Note 229). Thus we can be confident that it supports a wide range of radio astronomical observation modalities, including all those that are likely to occur with the EVLA. The MeasurementSet is based on *AIPS++* Tables and therefore inherits all the data storage advantages of the Table system.

5.5. **Parallelization.** The software costs will be driven by the complexity required to achieve these various growths predicted by Moore's Law. Since (non-optical, non-quantum!) CPU speed will presumably saturate at a few to 10's of GHz, parallel processing *may* be required to fulfil Moore's law in 2009. Hence we may well be required to program on moderately parallel (tens of processors) architectures. Investigation and development of low I/O, moderately parallel algorithms for non-embarrassingly-parallel problems must therefore be budgeted. In addition, the entire question of how best to parallelize I/O must be addressed since this may well be a limiting factor. These considerations apply equally to both desktop and central compute servers since both may well possess parallel architectures.

Relatively little work has been done on parallelizing radio astronomy imaging algorithms. The exceptions are an early use of PVM in multiple field deconvolution by Cornwell (VLA Scientific Memo 164, 1993), and more recently the parallelization project conducted jointly by NCSA and NRAO within the context of *AIPS++*. The latter project has the goals of (i) providing a common framework for developers to develop parallel algorithms, (ii) parallelizing key imaging algorithms, and (iii) providing the radio astronomical community with access to parallel facilities. An overview of this effort is available in the paper by Roberts. Some recent results on timing are available as *AIPS++* Note 232.

The parallelization project funds 4 positions at NCSA and NRAO, and provides *AIPS++* with excellent access to NCSA experts and resources. For example, we have recently ported *AIPS++* to the UNM Albuquerque High Performance Computing Center RoadRunner cluster. This latter facility is being used to develop and test algorithms such as a parallelized version of the *AIPS++* wide-field imaging algorithm.

In this memo so far, we have assumed that the various algorithms simply speed up in proportion to the number of processors available. This can be a valid assumption for the so-called "embarrassingly parallel" algorithms such as spectral line imaging in which the coupling between the processing required for different channels is minimal. However, for more complex algorithms such as multiple image deconvolution, the speed up factor may not be linear with the number of processors but may instead go as some power law, perhaps as bad as the square root. Development and testing is required to find parallelization strategies that yield high speed up factors for the expected number of processors. This of course is a key part of the parallelization project.

The NCSA/NRAO parallelization initiative is focussed around the use of the Message Passing Interface (MPI) to coordinate processing on multiple CPUs. An alternate and higher-level strategy has been pursued by the correlator group at DRAO in the ACSIS project. ACSIS is a digital auto-correlator being built at the DRAO to handle array heterodyne receivers on the JCMT. It uses a Beowulf cluster to provide the necessary processing for the peak data rate of 10.5 MB/s (about 40% of the EVLA peak data rate). *AIPS++* C++ classes and glish are used to implement a distributed object system that processes the data on cluster of loosely-coupled Linux-based Pentium III computers.

5.6. **Pipeline processing.** Pipelined processing must be supported in the reduction package. In addition, the necessary contextual information must be passed on from the scheduling software to the pipeline *e.g.* to designate the type of reduction to be applied to particular subsets of the data. This will require either addition of tags to the existing schedule format (easy) or the development of a new scheduling package (considerably harder).

The pipeline itself should support processing of the telescope data using scripts that are tuned to specific situations. For the simpler observational scenarios, these are nothing more than the encapsulation of procedures in the cookbook into executable scripts. For more complex observational scenarios, some scientific investigation and development will be necessary.

The scripting language used in a pipeline must allow variable substitution, functions, complex branching, process handling, and extensive processing of results within the scripting language itself. All of this is possible with the Glish language used within *AIPS++*.

The actual mechanics of pipelines require the use of *meta-information* about the observational. Procedures for handling meta-information within *AIPS++* are being developed as part of the ongoing pipeline development at NCSA.

5.7. **Conclusions.** *AIPS++* has been designed from the start to allow satisfaction of all of these demands. By comparison, satisfying these demands in AIPS (or another legacy package such as MIRIAD) would mandate new layers on software to be placed on top of the existing code base. Hence our strategy is to base our software efforts around *AIPS++*, bearing in mind that the current versions of AIPS and *AIPS++* provide a baseline of guaranteed capabilities similar to those available with the current VLA.

One potential concern is that *AIPS++* has not yet been widely adopted for the processing of synthesis data. Some early adopters have been using the package, as have dedicated bands of testers at various consortium sites. From this experience, we see two major obstacles to wide spread adoption of the package: the sheer complexity of the package, and some remaining inadequacies in the user interface and documentation. Neither of these two obstacles are likely to be permanent, and indeed we have ongoing strategies for addressing both issues. In no case have we found a flaw in the overall design of *AIPS++* that would prevent its eventual widespread adoption.

6. Summary

We have investigated the hardware and software needs for calibration and imaging for the EVLA.

We have used two different methods to estimate the processing hardware required to support observations with the EVLA. Both give an average CPU rate of $\sim 50 - 100$ Gflop, and a data rate of ten's of TB per year. Moore's law tells us that this would in principle be sustainable with the equivalent to a personal computer

[4]. Allowing a comfortable overhead of an order of magnitude argues for a parallel computer having 10's of nodes, for a cost of \$100K - \$200K. The general sense of these conclusions agrees with those found (Glendenning, private communication) for ALMA, where the output data rate is similar.

There are some important caveats to this conclusion. First, we note that to achieve these numbers, one may have to re-engineer some of the algorithms to be best suited to the computing hardware then available. The example that we looked at in some detail is wide-field imaging, where simply scaling the currently used multi-facet algorithm overestimates the computing required by about 1.5 orders of magnitude. Such work is ongoing as part of the NCSA/NRAO collaboration on parallelization of *AIPS++*algorithms. Second, we note that some new algorithms, such as those needed for radio frequency mitigation, will be required. Neither paths of development are particularly difficult but both must be followed.

We note that the throttle of the EVLA is essentially Moore's law: as Moore's law is followed with time (or not), we can contemplate the expansion of the capabilities of the EVLA. Crunching the numbers, we find that the maximum output rate of the WIDAR correlator would then be reached in about 2030!

On the software side, we have described the key requirements for any package that will support EVLA observations. A *guaranteed* level of processing is available with the existing AIPS and *AIPS++*packages. However, to fulfil the wide range of observational capabilities of the EVLA will require some continuing development.

NRAO, PO Box 0, Socorro, NM, 87801

[4]At this point, we feel that Machrone's Law should be mentioned: *the machine you want always costs \$5000*