CARMA memo xxx - 10feb2004

–draft– CARMA MIRIAD –draft–

a.k.a. MIRIAD V4, a.k.a. the ABC's of MIRIAD

Teuben, P.J. and Wright, M.C.H.
University of Maryland, College Park, MD 20742
Radio Astronomy laboratory, University of California, Berkeley, CA, 94720

ABSTRACT

We summarize the current status of MIRIAD, in particular efforts that have been going on since the introduction of CVS for code management in early 2001. Miriad is routinely used for BIMA, OVRO, ATNF and WSRT data reduction. In collaboration with WSRT and ATNF several modules for the analysis of data from these arrays have been added to MIRIAD. MIRIAD is being used for ALMA, ATA, CARMA, and LOFAR simulations. Support for more than 256 antennas and for datasets larger than 2GB (8GB upcoming) has been added to enable ATA and LOFAR simulations. Other new directions include the WSRT parallelization efforts and a python interface to miriad.

1. Introduction

After Sault departed Illinois in 1990(?), development of miriad continued in two separate versions: the A (ATNF, initially Sault) and B (BIMA, first at Illinois, then at Maryland). An attempt was made to merge the two versions, as presented in a proposal by Teuben at a BIMA board meeting in 2000 (?), but the ATNF group preferred to continue their own development. This proposal in turn was triggered by the departure of the then miriad maintainer Doug Roberts, which left no formal support for Miriad maintenance at NCSA. The flat tree and RCS-based checkin/checkout system was inflexible, and adding new files was very complicated. Early in 2001 the situation became critical (software rot etc.), and Maryland decided to take the RCS system and import it into CVS for a more modern approach to shared development. which was dubbed Version 3 of Miriad.

This version had a 2 year development cycle, which nominally ended February 2003, when the Version 4 branch was merged into the mainline development. In practice this meant that Version 3 would only obtain bug fixes for as long as we need it. In January 2004 Version 3 was closed, since data could now be reliably transferred between Miriad3 and Miriad4.

We now consider Miriad4 the "C" release (C for CARMA), which also signals the manual merging of as much ATNF specific code as can be managed, as well as Oosterloo's WSRT code (the initial impetus to solve the 2GB problem came from Tom Oosterloo at the WSRT where Miriad is now used routinely).

2. MIRIAD1 and MIRIAD2

Here is a brief chronology of the development of Miriad. Version numbers were not really as well established as they are from 3.0 onwards.

1987 Miriad ("V1") developed by Sault, largely based on his experiences with WERONG. Initially mainly dealing with visibilities, and a simple image interface. At that time GIPSY was thought to be the image analysis package, though fairly soon image analysis routines were added to Miriad.

1990 Bob Sault moved to ATNF, and together with Neil Killeen were responsible for a substantial portion of development, and a large number of new and improved tasks (some ATCA specific).

1990 Mark Stupar overhauls the directory structure, much to Sault's chagrin (the basis of the later split between the "A" "B" versions of MIRIAD. We call this Miriad V2.

1995 flat tree RCS archive in Illinois (Doug Roberts)

1997 mini (C) version of the miriad library, as used in AIPS++, was extracted. Later (see below) this was coded in ANSI-C, for portability. This "mini" library was code-wise identical to the one in miriad.

2001 Miriad V3 born out of CVS, coincides with the move of development from NCSA to UMD.

2003 Miriad V4 with files larger than 2GB and 8GB. Also dubbed the CARMA release.

2004 Miriad V3.1.1 (Released as 04jan13) was the last official Miriad V3.

3. MIRIAD3

Here is a list of the accomplishments of MIRIAD V3. We have release Version 3.1.1, which is now off a branch (-r MIR3) since MIRIAD V4 is now at the head of development. Only bug fixes are applied to this version of MIRIAD.

1. Miriad source code imported into CVS

2. Minor bug fixes, minor feature additions, and a few new program here and there (e.g. `deproject, rotcurmask`)

3. (some of it done before?) antenna could be 1..256, since baselines were counted (256*A1+A2). Already the ATA needed about 500, so the formulae was changed

4. fighting with the SONG team to keep the software in sync (not 100% successfull yet)

5. started to integrate ATNF specific code (therefor loosing the need for two versions of miriad?). We may have to keep up with some of the new things in the ATNF release, since they have expressed no real interest in merging the versions.

6. ...

4. MIRIAD4

In the following list only a few items have been implemented[1], most is just an incomplete wish list.

1. website needs to be simplified. currently on `http://www.astro.umd.edu/~teuben/miriad/` and only linked in via `http://bima.astro.umd.edu/miriad/` where it should be.

2. * pyramid: a simple python interface

3. * last remaining remnants of the old (baseline=256*A1+A2) formulue, and introduction of the new formulae that can handle > 2048 antennae. ???? Eric Greisen chimed in. Can't use > 256 due to random groups convention (well, maybe 2047?). But our fits program can't handle > 256 for now. Internally we can do 2048 (for now), though there is a suggested formulae to change this to 32768, the absolute maximum for an integer*4.

4. are we sure LFS is working for fits files?? needs to be tested (since AIPS++ just ran into that, i began to wonder too....)

5. MAXBUF could be bigger? better optimized? configure (see below) will do this eventually.

6. * The split MAXDIM/MAXDIM2, MAXANT/MAXANT2 is still not perfect... too many parameters always will result into situations wehre it didn't work.

   Also note there is some code for old vectorizing compilers/machines that use compiler directives like `c#maxloop 32`. For MAXANT > 32 obviously this will not do much good.

---

[1] listed with a *

7. export of small variables into type safe locations (as in NEMO's outkeys= interface).
8. should introduce an integer VERSION in the visibility data, just to make sure that data in the future remains in lock with the software
9. integer*8 support; ratty flag to go back to the old situation if the fortran compiler does not support integer*8 [this will support scratchfiles > 8GB]
10. write out proper long integers in itemize question is how fortran can do this?
11. a conversion routine from MIR4 to MIR3 format (int8 → int) if possible this program/script should check if all the int8 variables are short enough that they can be cut into an int4:

    ```
    nwcorr
    ncorr
    vislen (points to visdata, which does not need to be changed)

    ? flags ?? could this ever be > 2GB ???
    ? wflags
    ```

12. programs that read ascii files, tables and such should *as a rule* skip lines that start with the (universal unix) comment symbol (#)
13. mir-config that can spit out the cflags, libs etc... for example , in the current release uvio is not linked with LFS, unlike the rest. so uvio will hang....
14. * use scropen8,scrclose8,scread8,scrwrite8 in addition to scropen stuff. this will simplify the amount of code that needs integer*8
15. design a fftw interface to miriad's fft routines - should give an impressive speedup, certainly on intel hardware
16. the UVFITS random groups format should be replaced with the modern FITS IDI format (used by VLBA, AIPS++ ?, SMA, etc.). Also related to the 256 maximum number of antennae problem.
17. autoconf; see MIR/install for a start on this.
18. a better mir.test to excersize and certify a release

```
ATNF specific tasks:

atlod    Convert an RPFITS file into Miriad uv format.
atfix    Apply various miscellaneous corrections to ATCA visibility data
```

```
elevcor  Correct ATCA data for elevation-dependent gain
plboot   Set the flux scale of a visibility dataset given a planet obs.    ***


WSRT specific tasks:

OVRO specific tasks:
```

5. Data I/O

The Miriad Benchmark memo introduced a simple but workable benchmark to excersize the miriad (UV) I/O system to aid in understanding the scaleability. The `uvgen` task in particular can be estimated to calculate the amount of time it takes for a snapshot to be written to disk.

6. CVS

Here are some technical notes on CVS usage in MIRIAD, including branching, tagging and versioning:

```
Branching:  only done for serious development that has to happen in parallel to ongoing
            development, particularly when it's an experiment that may fail.
            (some of these can also happen in a persons private sandbox that never gets
     committed)

     cvs co miriad                # checkout a fresh copy, (tag it???)
     cvs tag -b ....              # branch it
     cvs update -r BTNAME

     ...
     cvs co miriad                # checkout another fresh copy
     cvs -j BTNAME                # merge it, resolved conflict, build it, test it, etc.etc
     cvs ci                       # check back all the changes

Tagging:    normally done when a version changes (e.g. from 4.0.2 to 4.0.3)

     Do this from a clean checkout (also to prevent that WIP and RAD gets tagged)

     cvs co miriad
```

```
cd miriad
edit VERSIONS         <- add the date that 4.0.2 is done, and mark 4.0.3 as "W.I.P"
edit cvs.tags         <- add 4_0_2, which is what you're about to do
cvs tag mir_4_0_2
edit VERSION          <- change to the new one, 4.0.3 in this case
cvs ci VERSION
```

7. Miriad 2003 telecon

7.1. Agenda

I see our goals as:

1. Determine what enhancements to the Miriad might warrant an increased investment in development.
2. Determine if there is sufficient need for a face-to-face meeting to coordinate such development.
3. Develop a prototype agenda for such a face-to-face meeting.

AGENDA

Below is a list of possible areas of development. I recommend that for each topic we attempt to briefly address the following questions:

- What is the motivation for pursuing this topic? How important is it?
- Would the topic require fundemental changes to the Miriad software infrastructure?
- Are their obvious paths to achieving desired capabilities?
- Does the topic require more discussion?

There are quite a few topics here, so we can't get into too much detail. (A need for detailed discussion is perhaps motivation for a face-to-face.)

Topic:

* 64-bit platform support
* Large file support (Miriad, FITS)
* Improved scripting with Python (enhancements to pyramid)
  - better integration with Miriad tasking

- flexible interrogation of uv and image datasets
- returning task output to scripting environment
- integration of non-task software

* Parallel Processing
  - need framework?,
  - transparent handling of platform dependence (e.g. parallel clean)
* Parallel I/O
* Improved Visualization
  - user's choice?
  - interactions with tasks, scripting environment?
* Algorithm development
  - support for heterogenous primary beams
  - multi-scale deconvolution
  - new calibration techniques: use of WVR, joint solving for polynomials in gain curves.
* General evolution of Miriad processing environment?
  If we imagine Python allowing for the integration of other software or other modes of processing, how does that change the way we look at Miriad as a processing environment? How do we plan for the future (e.g. multi-processing, GUIs, distributed processing)?

7.2. Minutes

Miriad Development Working Group Minutes
2003-06-11 Wednesday 1pm ET

Attendance:
MD: Peter Teuben, Stuart Vogel, Marc Pound
IL: Dick Crutcher, Ray Plante, Dave Mehringer
UCB: Mel Wright

GOALS

1. Determine what enhancements to the Miriad might warrant an increased

investment in development.
2. Determine if there is sufficient need for a face-to-face meeting to coordinate such development.
3. Develop a prototype agenda for such a face-to-face meeting.

AGENDA

Below is a list of possible areas of development. I recommend that for each topic we attempt to briefly address the following questions:

- o What is the motivation for pursuing this topic? How important is it?
- o Would the topic require fundemental changes to the Miriad software infrastructure?
- o Are their obvious paths to achieving desired capabilities?
- o Does the topic require more discussion?

There are quite a few topics here, so we can't get into too much detail. (A need for detailed discussion is perhaps motivation for a face-to-face.)

Topic:
* 64-bit platform support
* Large file support (Miriad, FITS)
* Improved scripting with Python (enhancements to pyramid)
  - better integration with Miriad tasking
  - flexible interrogation of uv and image datasets
  - returning task output to scripting environment
  - integration of non-task software
* Parallel Processing
  - need framework?,
  - transparent handling of platform dependence (e.g. parallel clean)
* Parallel I/O
* Improved Visualization
  - user's choice?
  - interactions with tasks, scripting environment?
* Algorithm development
  - support for heterogenous primary beams
  - multi-scale deconvolution
  - new calibration techniques: use of WVR, joint solving for polynomials in gain curves.
* General evolution of Miriad processing environment?

If we imagine Python allowing for the integration of other software or other modes of processing, how does that change the way we look at Miriad as a processing environment? How do we plan for the future (e.g. multi-processing, GUIs, distributed processing)?

MINUTES

Dick explained the motivation for this meeting (failure of AIPS++ to build a user community, examination of how to support high-end processing, ...). There were many comments about the need to be science driven.

Mel: On-the-fly, parallel processing will become increasingly important for future telescopes with high data rates (e.g. ATA).

Ray: We should assume that enhancements should not diminish current capabilities nor require a major change in how users use the package.

We assumed all gathered had copy of agenda and the draft memo, "CARMA MIRIAD". Peter indicated that the agenda was a more general list compared to the memo's wish list; the wish list items, thus, would be covered in the agenda.

* 64-bit Platform Support

There is a build on IL's Itanium box, thanks to Peter, using 32-bit compatability mode. True 64-bit support should be straight-forward and can be handled via compile flags. No further discussion needed.

* Large File Support

There are two limits: 2 GB & 8 GB. The latter requires integer*8 support in general and a minor change at the format level (dataset size pointer; see compatibility issues below). Solution is fairly well understood. Peter: some minor improvements supporting the EOD pointer within the uvio layer would be useful.

Compatibility Issues:

```
  * Miriad4 will be able to read either Miriad4 or Miriad3 datasets.
  * Miriad4 will convert automatically to Miriad4 datasets in any task
      that creates new output datasets.
  * Ray: Are there any compatibility issues for tasks that do in situ
      updates (e.g. uvflag)?
  * Miriad3 will be updated to at least detect Miriad4 datasets.  May
      attempt to deal with them if they are small enough; otherwise,
      fail.

No further discussion needed.

* Improved scripting with Python

Desired capabilities:
       - better integration with Miriad tasking:
            automatic generation of task interfaces as Python functions
            Ray has investigated how to expose necessary metadata to Python
       - flexible interrogation of uv and image datasets:
            to get info like how many channels in a window
            allows generic recipes to adapt to datasets
       - returning task output to scripting environment
            ex: getting stats from histo into Python variables.
            probably requires mods in task Fortran code
            could develop simple API that could be integrated into
              tasks piecemeal, as needed.  (not all tasks would need this.)
       - integration of non-Miriad task software
            e.g. using other Python modules like XML
                 accessing software from other astronomical packages.

These are important for the Pipeline but are useful to desktop users
as well.  None of these are particularly difficult, but it would
require additional coordination and, most importantly, input from
users.

Mel: Concern about how tightly we bind to Python.  What if something
     better comes along later?

Peter: SWIG will help with this: define binding SWIG's interface
     meta-language, SWIG then can generate bindings for various
     supported scripting languages.
```

* Parallel Processing

** Parallel I/O:

Two approachs:
1. Mel explained that parallel I/O can often be effectively achieved by strategic splitting of datasets, primarily by channel. This solution requires no change to the format. Possible downside is whether this rough cut approach limits the amount of effective parallism that can be achieved (compared to something more fine-grained). Do we forsee splitting data in ways other than channel?

2. An alternative would be to convert Miriad datasets (Miriad5?) to HDF5 format with transparent support for parallel I/O. This format is well-supported and widely used, and substantial expertise resides at NCSA. The downside is that it could have substantial costs in rewriting the Miriad I/O layer and to users in supporting backward compatibility.

We recommend a two-pronged prototyping approach: Test out how far we can go with the rough-cut approach, and to do a study in collaboration with the HDF group to understand how Miriad data might be stored in HDF5.

[Ray notes: we expect to start prototyping the former approach in the Pipeline. It makes sense to do the latter at NCSA as well.]

** Parallel Processing

What strategies do we want to apply? DIY embarrasingly parallel (EP) at the script level? fine-grained parallelization with MPI?

What examples are there of parallelization that isn't fundementally channel-based? Mosaicing (of continuum sources), Gridding?

Ray: We have experience with MPI (from AIPS++), but we would like to push the EP approach as far as we can before considering MPI. MPI is more intrusive to the fortran layer and is ultimately less flexible to running in different parallel environments (e.g. clusters, Condor

pools, and service-based distributed computing).

* Improved Visualization

Ray: Is visualization adequate in Miriad? Have users been sufficiently successful with current available choices (Karma, DS9, Aipsview, Miriad visualization tasks)? Where is interaction with tasks and scripting environment important (e.g. creating clean boxes)? Is it sufficiently well supported?

Peter: Less publicized capability for creating clean boxes using DS9.

Peter: Current assessment of support for Miriad
  DS9 -- image data piped from Miriad (using DS9's XPA mechanism)
  Karma -- reads Miriad natively (or via FITS?)
  Aipsview -- currently supports FITS and Miriad (via AIPS++ lib)
  IDL -- via FITS

Marc: we should not invest in re-implementing what already exists with current apps.

Peter: one area that is a bit crude still, IMHO, is WIP. There already exists a Python binding to PGPlot, so perhaps there is an opportunity to improve things.

Conclusion: no fundemental changes necessary. Incremental improvements to enable new functionality using one or more of the existing viz options may be helpful.

** IDL Support

There was a discussion thread on the prospects of supporting IDL. Since there are a number of people within our community that use IDL, there may be some benefit.

Peter: have resource at UMD in Stephen White.

Ray expressed concern about getting into a situation of supporting duplicate functionality in IDL and Python.

Peter did not think that there would be significant demand for manipulating uv-data in IDL; however, there would certainly be interest in reading Miriad images. Given IDLs existing image display and manipulation capabilities, supporting IDL need not be more than providing a Miriad image reader.

* Algorithm Development

Dick: Question: How capable is Miriad as a development environment?

Mel: Experience has shown that it is quite capable as new tasks have been submitted by a variety of people. The typical pattern for development is to take an existing task and modify it or use it as a template.

Peter: The processing model is not as general as the Measurement Equation formalism used by AIPS++ (as the ME didn't exist when Miriad was developed). There are some aspects of the library that might be considered expert knowlegde, but the capabilities are all there.

Ray: Two philosophical points (i.e. they may be important enough at this point to drive any changes to Miriad infrastructure):

1) The development pattern of modifying existing tasks may limit one's ability to try something fundementally different.
2) One of features of AIPS++ that was meant to make new development easier is the exposing of low-level data and algorithm access to the scripting layer; this allows people to try out ideas in a rapid development environment before investing in a C++ implementation. We are not envisioning supporting this in Miriad.

Specific Capabilities:
** Support for heterogeneous primary beams

Ray: what currently exists?
Mel: The basic functionality exists pb* routines support this.
Primary beams are usually pre-defined. There is one that is the

geometric mean of an OVRO and BIMA beam; nothing currently does this combination automatically. Could be improved.

Dick: What about the case of different antennas looking at different fields?

Mel: Initially we thought about allowing for this; however, after some thought one realizes that there are a host of complicated, systematic errors one has to deal with. The effort for handling these correctly may not be worth any increase in observing speed.

** Multi-scale deconvolution

Mosmem attempts this in its own way; it's slow.

GILDAS has this functionality; can we use it (e.g. call it from Python?)

The AIPS++ version is based on the GILDAS idea; however, it is implemented differently. Its implementation uses the standard AIPS++ decon tools to do the deconvolution on different scales. The coordinatation and combination of subsequent models are done in the scripting layer (glish).

Conclusion: there is user interest, but there are several approaches that could be tried.

** New Calibration techniques

use of WVR...
Mel: when this was prototyped before, WVR data were converted to corrections stored in a gains item. Straight-forward from there.

using optical pointing offsets to correct pointing errors...
Mel: best done on-line

decorrelation corrections...
[more words?]

Conclusion: while there do not seem to be any critical path capabilities needed for CARMA, there is room for improvements and experimentation with new techniques.

* General Evolution of Miriad Processing environment.

** GUIs

Peter: experience shows that users want scripts more than GUIs. Python would allow users to experiment with simple GUIs. Some people have created personal, custom GUIs as convenient way of running scripts with different inputs. A little bit of glue for describing Miriad task (and script) inputs would make it easier to create GUIs.

Mel: GUIs consider imp. for ALMA, according to the requirements. GUIs are helpful for guiding new users through a process; this is considered an advantage of GILDAS.

Ray: There are different types of GUIs:

1) High-level, user-oriented GUIs dedicated to a specific sequence of processing steps; often refered to as "wizards". This level was never achieved with AIPS++, though highly desired by users, according to their feedback.

2) General workflow GUIs. This is an integrated GUI environement for stitching together and launching tasks. AIPS++ had this, but it was not well-liked.

3) Custom GUIs: highly specialized interfaces to a particular task. xcorf is an example. These are often costly to build.

4) Quick-custom GUIs: these would be the simple GUIs whipped up by users, as Peter described.

Given our experience with Users, there is not a lot of demand for 2). Some experimentation could be done with 1) as needed. GUIs of type 3) will be written whenever needed, but not as a general practice.

I've started looking into the kind of glue Peter referred to

```
      as part of improving the integration of tasks in Python; it
      would also make building GUIs easier.

Conclusion: the main priority right now is to provide just enough
support to allow experimentation (by both users and developers) in
Python.  Any coordinated effort toward GUIs should be user-driven.

CONCLUSIONS/ACTION ITEMS

We have identified some areas that could use further developement.
For most of these, we have a sense of what level of effort is
warrented.  We have decided to collect these ideas and incorporate
them into Peter's draft note, transforming it into a white paper.
This will be distributed to our user community to get feedback.

We will proceed to work on the white paper over email and possibly
another telecon.  It is likely that we would postpone a face-to-face
meeting until after the white paper is prepared and some feedback
gotten from users.

Is Peter willing to be white paper editor?  Ray expects to be an
active contributer.

Action Items:
  Ray:    type up minutes and distribute.                Done.
          collect corrections/additions
  Peter:  file minutes into rough draft in some form
            to start transformation to white paper.
```

References

[1] WERONG - Sault's thesis? 1986?

[2] Sault R.J., Teuben P.J., Wright M.C.H., 1995, "A retrospective view of Miriad" in Astronomical Data Analysis Software and Systems IV, ed. R. Shaw, H.E. Payne, J.J.E. Hayes, ASP Conf. Ser., 77, 433-436.

[3] Teuben - BIMA memo 81 on miriad benchmarks (a "live" document)