The DiFX Software Correlator at the Institute of Radio Astronomy *Version 1.0*

J.S. Morgan
INAF – Istituto di Radioastronomia,
Via P. Gobetti, 101
40129 Bologna, Italy.
`jmorgan@ira.inaf.it`

October 31, 2008

Contents

Chapter 1

Introduction

This report documents the software correlator and associated software that we have installed at the Institute of Radio astronomy in Bologna. This will allow us to do our own Very Long Baseline Interferometry (VLBI) entirely within the institute all the way from antenna to publication. In addition it means that there will be expertise on correlation within the institute.

The centre of this effort is the DiFX software correlator which allows software correlation on a standard computing cluster. There is a range of software related to this correlator; we need to choose which software to use, fill any gaps in the pipeline, and work out how to coordinate these applications.

This guide is intended both as a user guide and as a repository for all the information required to run correlations within the institute. Particular effort has been put into making sure further sources of information are referenced within this guide.

1.1 Aims

In order to produce a pipeline which is as useful as possible we have kept in mind the following aims:

Compatibility

In order for our pipeline to be of the most use to our users, it should be compatible with the tools that IRA astronomers are most familiar with.

For this reason we have tried to make it compatible with the tools used by the European VLBI network (EVN) [1] since this is the VLBI network that our astronomers and antennas interact with the most. The main consequence of this is that we must use VEX files, which are currently used by the EVN but *not*, for the moment, by the VLBA.

As well compatibility with existing scheduling tools and data formats, we should also consider incompatibility with data reduction packages (see pipelining below).

Flexibility

Since one of the main advantages of software correlators is their flexibility, they will often be used for unusual projects which would be more difficult to process using a hardware correlator, or that may require correlation several times with different parameters. It therefore makes sense to make sure that the pipeline we develop is as flexible as possible too. We will try to be compatible with as many different input and output formats as possible.

Automation

Correlation requires the control of a large amount of data, files and parameters. It is important to keep the workload down, and keep errors to a minimum by automating as much of this process as much as possible. However, we have tried to keep all of the underlying tools accessible to maximise flexibility.

Pipelining

One advantage of a software correlator is that we can use the computer cluster for further data reduction once the correlation is complete. Therefore we want the maximum amount of integration with data reduction packages such as AIPS and casa.

1.2 Conventions

In this document:

- 'CPU core' refers to a single core of a multi-core CPU, or a single-core CPU.
- 'Processing core' refers to a number-crunching process of the DiFX software correlator ('core' in the DiFX instructions).
- 'Antenna' refers to a single station of an interferometer network.
- 'UV data' and 'UV dataset' refer to the correlated data.
- 'UV coordinates' and 'UVW coordinates' refers to the data recording the positions of the antennas in the UV plane.

- ‘Scan’ refers to a single pointing of the antennas.
- ‘Subband’ refers to what AIPS calls an IF.

All references to computer files or executable programs are written in `typewriter` font. AIPS, CALC/SOLVE, AIPS TASKS, FITS and VEX are in SMALL CAPS.

Chapter 2

Overview of the Correlation Process

In this chapter we present an overview of how the correlation is carried out with particular emphasis on the input and output files. For an overview of an entire VLBI observation from scheduling to imaging see [2, §5].

Correlation is the process of multiplying and accumulating (time-averaging) the waveforms from two or more antennas in an interferometer array [3, Ch 3.2].

In order to produce a correlator system which can replace existing hardware correlators we need to provide other functionality. Effectively, we need to be able to transform digitised broadband data (stored in a format such as Mark5) into a UV dataset (in FITS format).

The work which our pipeline does can broadly be split into four functions:

- Parsing the input files and presenting the data in a form which the correlator and geometric modelling tools can understand.
- Calculating the geometric delays and UVW coordinates of the antennas throughout the observation (geometric modelling).
- Multiplying and accumulating the broadband data to produce UV data (the correlation itself).
- Transforming the correlator output and associated data into a form which can easily be read by data reduction packages.

2.1 Inputs

2.1.1 Broadband Data

The most voluminous data is the broadband data itself.

It consists of a digitised waveform recorded at a high sampling rate, but with a low number of bits per sample (1 or 2 is most common for VLBI). It is typically produced at a rate of between 64Mbit/s and 2Gbit/s, producing of the order of 1 TB per station [4, 5, Whitney, Campbell, Tzioumis].

At the moment we are typically working with Mark5A [6], however the software correlator also supports other input formats (these have not been tested here).

2.1.2 Vex Files

VLBI Experiment files (vex) files are designed to "prescribe a complete description of a VLBI experiment including scheduling, data-taking and correlation" [7]. They contain all of the parameters necessary to carry out the observation.

When scheduling the observation, the astronomer produces a vex file which is then sent to each of the stations. The following are the parameters relevant to the correlation (or required to produce a valid fits file) contained in the vex file (either implicitly or explicitly).

- A schedule of the various scans (including, of course the start and end times of the observation).
- Names and coordinates of the antennas (including axis offset and mount).
- Names and coordinates of the sources.
- Subband frequencies, bandwidths, and sidebands.
- Formatting of the broadband data (including fanout and quantisation bits).
- Polarisations observed.

In order to "prescribe" a complete description of the experiment, parameters which cannot be known exactly prior to the observation must be included such as earth orientation parameters and clock offsets. However the vex file can be processed after the observation in order to add these.

Many of the parameters contained within the vex file can also be obtained from other sources, for example the coordinates of the antenna (which have a derivative due to tectonic plate movement). For the time being we have chosen to take all parameters from the vex file where possible, leaving the responsibility for errors with the astronomer.

2.1.3 Earth Orientation Parameters (EOPs)

These parameters describe how the orientation of the earth deviated from a standard model at the time of the observation [8, 9, III.D]. They are not required in order to correlate the data, however they are needed to produce the FITS-IDI file. The parameters can be downloaded from the internet. The uncertainty in the values decreases steadily for around a month after the epoch of the observation.

2.1.4 Antenna log files

A log file is generated by the field system [10] at each of the antennas. These files contain such data as the clock offset and rates; pcal [3, Ch 9.4] and system temperature (Tsys) measurements[3, Ch 1.2]; and comments made by the observer. The only ones relevant to the correlation process itself are the clock offsets[1]. However other parameters must be parsed and and passed on to the end-user.

2.1.5 Clock Offsets and Rates

The local oscillators of the antennas are driven by some frequency standard – usually a hydrogen maser. These frequency standards are chosen for their stability over the time range of the observation rather than absolute accuracy [3, Ch 3.2]. Throughout the observation the maser is compared with GPS time. The accuracy of any individual measurement is very low, ideally measurements should be taken over 24 hours, but often the log files only contain data over the time of the observation [2]. Using the values within the log files to obtain a clock model is usually sufficient, but certainly not optimal [11].

The correlator models the clock as having a simple offset and rate. Future versions of DiFX will characterise the maser with a polynomial.

2.1.6 Correlation Parameters

There are a few correlation parameters which are not recorded in the VEX file, which are chosen by the user. These are:

- The number of channels per subband
- The integration time

[1]In addition a single Tsys value can be given for each antenna in the software correlator. However as in the NRAO-DiFX pipeline we set these to 1.

[2]It should also be noted that the offsets for EVN antennas are also available online e.g. `http://www.ira.cnr/it/vlb_arc/gps/dec07/gps.ef`

These are currently manually entered into the correlator input file just before correlation.

There are many other parameters in the correlator input file which control various technical parameters, however we have always left these at their default values.

2.2 Geometric Modelling

In order to correlate the data correctly the position of antennas with respect to the source and to each other must be known extremely accurately. Specifically the "geometric delays" must be calculated [9, §III]. Essentially delays are calculated which, when applied to the broadband data, shift them in time so that they are aligned as if all the antennas were on a plane perpendicular to the source, passing through the centre of the earth. Thus for ground-based VLBI the delays will be between 0 and r_{earth}/c. A separate software package is used to calculate these data over the duration of the observation which are then provided to the correlator as ASCII tables. At the same time the UVW coordinates of the antennas and a simple atmospheric model are calculated. These are included in the final output file.

2.3 Correlation

Although this is the computer-intensive part of the process, mathematically the process is very simple[3].

1. The baseband data from every antenna are aligned with every other using the geometric delays calculated [12, §2.1.1],

2. The differing velocities of the station with respect to the source are accounted for (fringe rotation) [12, §2.1.2]

3. The baseband data from every antenna of every subband are channelised (fourier transformed) [12, §2.1.3]

4. The baseband data from every antenna of every subband are cross multiplied [12, §2.2.1]

5. The results of these cross multiplications are accumulated (integrated over time) [12, §2.2.2]

6. The resulting output is stored on disk.

[3]This very brief outline is somewhat specific to the DiFX software correlator. For a more general treatment see [9, §II.B]

2.4 Conversion

Finally the output of the correlator must be converted in to a form which can be read by data-reduction packages. There is also additional data from the log files which must be passed on to the end-user.

System Temperatures (Tsys)

The Tsys values taken throughout the experiment can be provided to the user either as a table in the FITS file or in an ANTAB file. This is a file which contains the gain curves of the antennas, and the system temperatures observed throughout the experiment. It allows the user to edit the data manually before reading it into the data reduction package.

Flagging

The log file may also record the time for which the antenna was off source or other problems. This can be provided to the user either as a table in the FITS file or as a UVFLG input file. As with the ANTAB file, providing a UVFLG file allows the user to perform manual editing if they wish.

Other

Log files may also include weather conditions throughout the observation.

2.4.1 FITS-IDI

The FITS Interferometry Data Interchange convention [13] is the standard interchange format for correlated VLBI data.

The Tsys values can be added to the FITS file directly, or they can be placed in an ANTAB file for later addition to the data once the values have been edited manually.

The following tables may be written to the FITS file (many are optional).

Table	Data
AG	Array Geometry
SU	Source
AN	Antenna
FR	Frequency
ML	Model
CT	EOPs
MC	Model Components
SO	Spacecraft Orbit
UV	UV Data and UV coordinates
TS	System temperature
PH	Phase Cal
WR	Weather
GN	Gain Curve

Table 2.1: Tables written by difx2fits

Chapter 3

The Software Correlator and Related Software

In this chapter we describe the various software which is available to carry out the steps described in the previous chapter. In chapter 4 we will describe the software we have chosen in more detail.

3.1 The Software Correlator

The heart of our correlator is the DiFX software correlator itself, developed at Swinburne University of Technology [12]. It essentially allows correlation to take place on any standard computing cluster.

Another similar effort is being coordinated by JIVE [14].

3.2 Geometry Modelling Software

CALC/SOLVE [15] is an well-established piece of software, maintained by the NASA Goddard Flight Center. This software is widely used in VLBI to calculate the position of the antenna relative to the source for the duration of observation. Two pieces of software `calcif` (part of the NRAO-DiFX pipeline) and `vex2model.pl` (part of vex2difx) are available which interact with CALC/SOLVE to produce the necessary data in tabular form. Different versions of CALC/SOLVE are available. In addition, there is other software which could in principle produce the necessary ASCII tables.

3.3 NRAO-DiFX

The aim of the NRAO-DiFX pipeline [16] is to provide the extra tools needed to allow the DiFX correlator to replace entirely the hardware correlator currently used for the VLBA. Once it is finished therefore, it will represent a complete and reliable system for software correlation.

The NRAO-DiFX distribution contains lots of tools associated with the software correlator, as well as a version of the software correlator itself, modified to read from Mark5 units.

While not yet at the production stage, our experience has been that the NRAO-DiFX pipeline is already reliable and comprehensive with clear instructions for use and installation.

The only thing which (from our point of view) is lacking in the NRAO-DiFX tool set is compatibility with our input files (namely VEX files and our antenna log files). In addition, since it is designed to replace a hardware correlator in a well-established system, it is currently built with less flexibility than we are aiming for.

3.4 vex2difx

`vex2config.pl` and `vex2model.pl` are a set of perl scripts for producing the input files for DiFX from VEX files. `vex2config.pl` produces the `.input` file, while `vex2model.pl` runs CALC/SOLVE and generates the ASCII tables.

3.5 IRA-DiFX

Having used all the available tools from these three sources, there were still a few gaps in the pipeline. These have been written in-house in python.

Chapter 4

Chosen Third-party Software in Detail

In this chapter we discuss the third-party software we have chosen for our pipeline. Refer to figure 4 for an overview.

4.1 `mpifxcorr` (DiFX)

Input files	Output files
`.uvw`	`.difx`
`.delay`	
`.input`	
`.machine`	
`.cores`	

`mpifxcorr` is the DiFX software correlator itself [12]. We are using the original version [17] rather than the NRAO-DiFX version. It is written in C++ and is built around two external libraries.

4.1.1 External Libraries

MPI

The Message Passing Interface (MPI) is a widely used API for running software in parallel on more than one computer [18]. It facilitates the launch of independent processes on a single machine, or on different machines in a cluster. Communication between the processes is achieved via rsh or ssh.

The implementation we have used is mpich [19]. A separate installation of mpich is maintained for exclusive use with the software correlator.

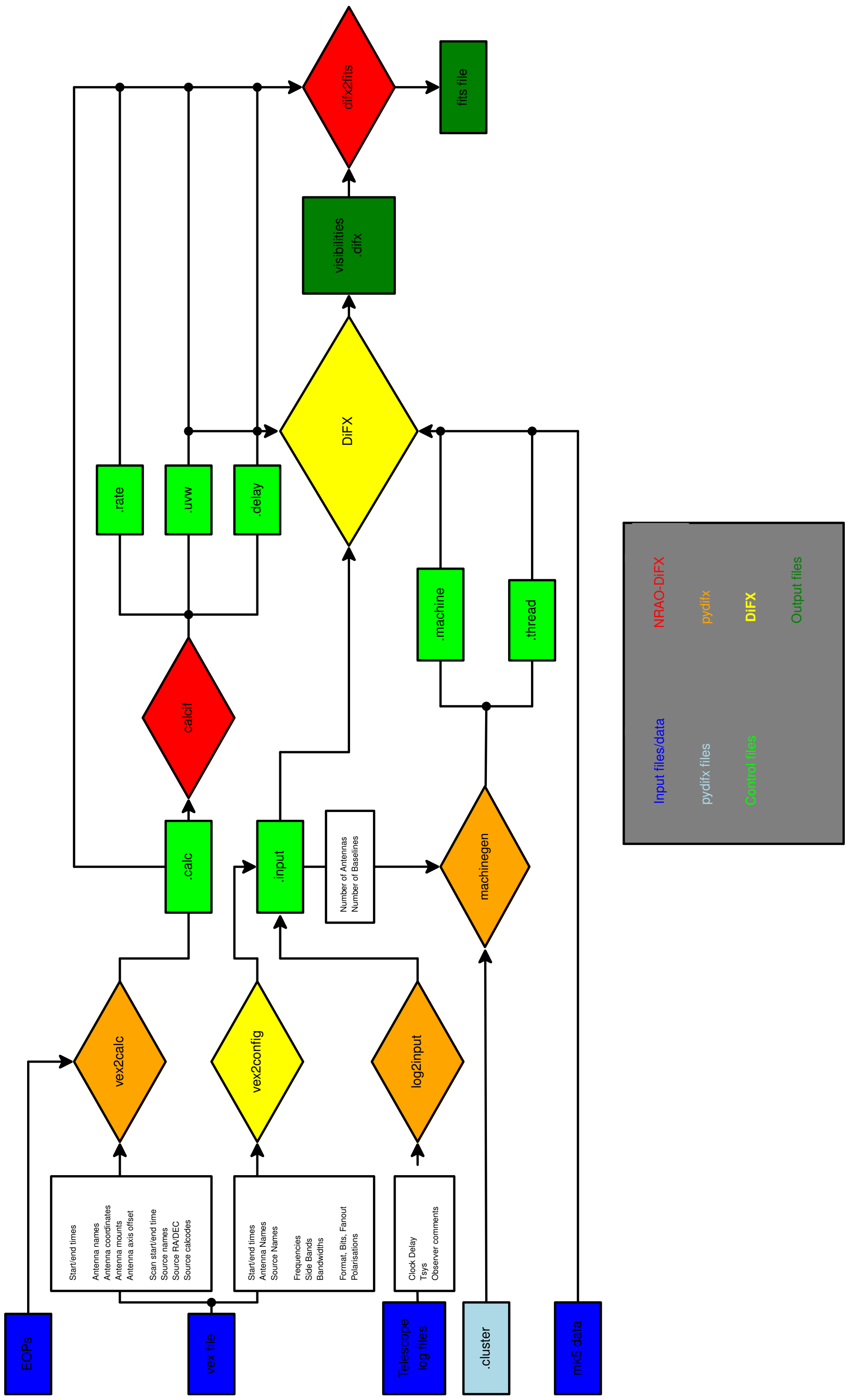

Table 4.1: Overview of the Correlator Pipeline

Intel® Integrated Performance Primitives

This is a closed-source library [20] which allows the use of optimised routines for the vector calculations which lie at the heart of the software correlator. In principal this library could be replaced by another vector library such as the open source 'AMD Performance Library' (APL) [21].

4.1.2 Structure of DiFX

The processes running on the cluster are split into 3 different types [12, §3.2].

FXManager

The first process is designated the FXManager. This manages the correlation and writes the final data to disk.

Datastream

One process per antenna is designated a Datastream. This node must be able to access the broadband data for that antenna. The data is read from file, unpacked, and chunks are passed onto the cores.

Core

All remaining processes are given the slightly ambiguous name of cores (herein referred to as processing cores). These processes fringe-rotate, channelise, correlate and accumulate the data. The resulting data is passed back to the FXManager for writing out to disk.

4.2 `vex2config.pl` (vex2difx)

Input files	Output files
`.skd` (vex file)	`.input`

We use `vex2config.pl` to produce the `.input` file for the correlator.

4.3 `calcif` (NRAO-DiFX)

Input files	Output files
`.calc`	`.uvw`
	`.delay`
	`.rate`

calcif is the program which interacts with CALC/SOLVE. A .calc file is the main input. calcif interacts with CALC/SOLVE via CALC server (which may run on a separate machine). The three output files are all simple ASCII tables.

- .uvw contains the UV coordinates of the antennas for the duration of the observation.
- .delay contains the geometric delay of each of the antennas for the duration of the observation.
- .rate contains a simple atmospheric model.

The .rate file is not used for the correlation but adds a table to the final FITS file. The .uvw and .delay files are used by DiFX. All three are used by difx2fits.

4.4 difx2fits (NRAO-DiFX)

Input files	Output files
.difx	.FITS
.uvw	
.delay	
.rate	
tsys	
pcal	
flags	
weather	

This program takes a variety of inputs, the most important being the correlator output, and produces a FITS-IDI file.

Some of the input files are optional.

Chapter 5

IRA-DiFX Software in Detail

In this chapter we describe the software that has been developed 'in-house'.

Only the the more important scripts are mentioned here. More extensive documentation can be found in the files themselves. All files have a docstring at the top. All of the executables have a docstring which will be displayed if they are run without arguments.

5.1 Vex parsers

5.1.1 `vex2calc.py`

Input files	Output files
`.skd` (VEXfile)	`.calc`

The main thing missing is some way to generate a `.calc` file which is required both by `calcif` and `difx2fits`. There is an official C library for interacting with VEX files, which we could have used. Instead we found it easier to write a VEX file parser in python. The disadvantage to this approach, is that it may not work on a particularly unusual VEX file, however extensive testing on the VEX files taken from the EVN archive did not throw up any major problems.

5.1.2 `vex2flag.py`

Input files	Output files
`.skd` (VEX file)	`flag`
	or
	UVFLG input file

The software correlator currently tries to correlate for every time between the start and the end of the correlation. This leave a small amount of nonsense data between the scans. This script parses the VEX file and produces a flag file which ensures times between the scans are flagged. `vex2flag.py` can generate an input file to `difx2fits` or an input file to the AIPS task UVFLG.

5.2 Log parsers

5.2.1 `log2input.py`

Input files	Output files
`.log`	`.input`

The DiFX `.input` file describes each station clock as having an offset and a rate.

This script reads all of the GPS information from the log files and calculates the intercept (offset) and slope (rate) of the line of best fit.

5.2.2 `log2tsys.py`

Input files	Output files
`.log`	`tsys`
	or
	ANTAB

`log2tsys.py` takes the Tsys readings from the log table and tabulates them. Using this output it is relatively easy to create an ANTAB file by hand.

5.2.3 `log2comment.py`

Input files	Output files
`.log`	ANTAB

`log2comment.py` takes the observer comments from the log table and prints them to stdout, correctly formatted for adding to an ANTAB file as comments.

5.3 Utilities

5.3.1 `machinegen.py`

Input files	Output files
`.input`	`.machine`
`.cluster`	`.threads`

Because number of datastream nodes depends on the number of antennas, the `.machine` file may change depending on the number of antennas. For this reason we need to define a cluster file which specifies the names of the machines in the cluster and how many CPU cores each machine has. From this information the `.machine` and `.thread` files can be generated.

5.3.2 `killdifx.py`

Input files
`.machine`

This small script simply connects to all the machines in a machine file and uses the `killall` utility to kill any zombie `mpifxcorr` threads which may have accumulated.

5.4 Wrappers for 3rd-party Programs

There are several purposes for these programs.

Firstly they simplify the running of `calcif`, and `mpifxcorr`; starting the CALC server in the case of the former, and starting the MPI job with the appropriate parameters in the case of the latter.

Secondly they provide functions with which pydifx (our python scripting class) can call these programs.

Thirdly they log the output of these programs both to stdout and to a log file. This is essential in the testing stages to allow debugging after the fact, and also to allow benchmarking. The standard python logging framework is used [22].

5.4.1 `calcif.py`

`calcif.py` checks that the CALC server is running and working, then runs `calcif`.

5.4.2 `difx2fits.py`

`difx2fits.py` runs `difx2fits`.

5.4.3 `mpifxcorr.py`

`mpifxcorr.py` runs some sanity checks on the input files and output files, calculates the correct number of processes and launches the MPI job.

5.5 Integration of the various tools

We need some way to tie all these separate software tools together. We are also concerned about giving the user the ability to automate and customise the correlation process using a scripting language.

5.5.1 `correlator_defaults.py` and `observation.py`

The `correlator_defaults` file contains default values for various parameters which are expected to remain the same across all correlations.

`observation.py` is placed in the correlation directory (i.e. the directory which contains all of the input and output files). It can be used to set values specific to the individual correlation.

5.5.2 `pydifx.py`

A simple python class pydifx has been written which provides a simple framework for scripting correlations in python. By writing a simple script it is simple to carry out the following operations:

- Run any of the tools above.
- Edit the input files.

In addition, all the normal python functions are present for renaming/moving files etc. AIPS can also be called (via ParselTongue [23]) from the same script. This is particularly useful for debugging and benchmarking. An example script is given in appendix C.

Chapter 6

The Computer Cluster Used for Correlation

The correlator is installed on our experimental cluster, which is also an experimental grid node. We are aiming for the cluster to act as an all-purpose machine for radio astronomy data reduction.

6.1 Machines

The cluster is heterogeneous: Consisting of 8 machines connected with a 1 Gbit switch (Table 6.1). This 24 CPU cluster represents a minimum amount of computing power required to correlate sensible amounts of data.

Table 6.1: Cluster used for the correlation

Machine	Number of Cores	Type of Processor	Clock Speed
wn01	2 (2 × single core)	Intel 4	3 GHz
wn02	2 (2 × single core)	Intel 4	3 GHz
wn03	2 (2 × single core)	Intel 4	3 GHz
wn04	2 (2 × single core)	Intel 4	3 GHz
wn05	4 (2 × dual core)	AMD Opteron 270	2 GHz
wn06	4 (2 × dual core)	AMD Opteron 270	2 GHz
wn07	4 (2 × dual core)	AMD Opteron 270	2 GHz
wn08	4 (2 × dual core)	AMD Opteron 270	2 GHz

6.2 File Systems

The input data is currently stored on a GFS file system connected to the 1Gbit switch. The output is stored on the same drive. There is also some local storage space on each of the machines.

6.3 Benchmarks

For an observation with 4 antennas, and a modest bandwidth of 128 Mbit/s we are able to correlate 3 minutes of data in 10 minutes.

6.3.1 Bottlenecks

If the number of channels per subband is set sufficiently high then the CPUs are utilised 100%. Reducing the integration time seems to have little effect on the speed of correlation.

However if the number of channels is small, then neither the network nor the CPU are at capacity and it is not clear what is the limiting factor.

For the low bandwidth data which we are using, making the Mark5 data accessible locally rather than via NFS seems to make little difference.

We are planning to use our benchmark script on other clusters to understand in more detail the limiting factors.

6.4 Other Software

We have also installed AIPS and ParselTongue (a python interface to classic AIPS) on the same cluster. We have already experimented with creating scripts which run the correlator and AIPS (via ParselTongue[23]) from the same script. We have also installed the EVN pipeline [24] used at JIVE to automate the first stages of data reduction (using ParselTongue).

Chapter 7

Correlation: Step by Step

This chapter presents an extremely brief overview of the steps required to carry out a correlation. A little more detail is given in Appendix A.

7.1 Preparing the Input Files

1. Process the VEX file to create the `.input` file using `vex2config.pl`[1]
2. Edit the `.input` file by hand
   (a) Remove any unused antennas (alternatively this can be changed in the VEX file
   (b) Add paths to the broadband data
   (c) Set the number of channels and integration time for each source
3. Process the VEX file to create the `.calc` file using vex2calc.py
4. Edit the `.calc` file by hand
   (a) Remove any unused antennas and (optionally) any unused scans
5. Process the log files to add the clock data to the `.input` file using `log2input.py`
6. Process the log files to produce the ANTAB file and/or `difx2fits` input files
7. Create the `.machine` and `.thread` file using `machinegen.py`

[1]`vex2config.pl` is rather unreliable. Another tool which can be used is `vex2difx` from the latest version of NRAO-DiFX. The VEX file produced by either of these tools will require extensive editing by hand.

7.2 Geometric Modelling

1. Run `calcif.py` to generate the `.uvw` `.delay` and `.rate` files

7.3 Correlation

1. Run the correlator (`mpifxcorr.py`)

7.4 Conversion and Post-processing

1. Next we run `difx2fits.py`
2. Copy the FITS file to an appropriate directory
3. Read the data into ANTAB using FITLD or run the EVN pipeline

Chapter 8

Upgrade Path

The existing tools outlined here will all be maintained and this will be considered a stable installation. Soon we plan to upgrade the correlator, as well as installing a development version. This will give us a total of three software correlators installed concurrently.

8.1 Next Upgrade

NRAO-DiFX 1.1

The various NRAO-DiFX tools that we are working are taken from various intermediate versions between 1.0 and 1.1 We will upgrade to the 1.1 version [25].

We will also install the NRAO-DiFX version of the software correlator itself.

AIPS

Some small upgrades have been made to the latest version of AIPS (31DEC08) specifically for use with NRAO-DiFX. We will upgrade to the latest version to take advantages of these changes.

ParselTongue

Recently a new version of ParselTongue (1.1) has been released with some developments related to parallel execution.

Casa

Casa [26] is not yet used widely for VLBI (if at all), however we want to make our data reduction cluster as general as possible. Casa may prove useful for wide-field VLBI imaging where numbers of channels per subband may exceed built-in limits in AIPS.

OpenMPI

Some users of DiFX have reported increased performance using OpenMPI [27] rather than mpich.

IPP

We are currently running the same version (5.3) of IPP on all our machines. We will probably see a significant increase in performance by running the 64bit version on our 64bit machines. We should ensure that we always have the latest version installed.

EVN "Pypeline"

The EVN pipeline [24] is already installed and working, We may wish to customise it.

8.1.1 Other Changes

More Machines

There are other machines in the institute which are connected to the 1 Gbps switch, and could be used to increase the performance of the cluster.

Users and Groups

We should give some thought to how we will organise permissions for the input and output files, and which users should run jobs.

File Systems

We are continuing to experiment with parallel file systems to store the large amount of input data.

8.1.2 Development Branch

NRAO-DiFX 2.0

At the same time we will install the development branch of the correlator: NRAO-DiFX 2.0 [25, §2.4]. This will probably become the standard correlator installed at all the sites currently using DiFX. This is largely because NRAO-DiFX 2.0 will use VEX files making it compatible with the systems used in other institutes.

8.1.3 Our Contribution

At the last DiFX conference in Bonn we agreed to work on two aspects of NRAO-DiFX 2.0.

Logging

Adapting the python logging system to the status broadcast system (difxmessage) of NRAO-DiFX 1.1 [25, §10].

Benchmarking

Using pydifx, we will collaborate with Cagliari to produce some benchmarking scripts for the software correlator.

Acknowledgements

Many thanks to Franco Mantovani, Mauro Nanni, Steven Tingay and Walter Alef for advice on many aspects of informatics, correlation and interferometry.

This research was supported by the EU Framework 6 Marie Curie Early Stage Training programme under contract number MEST-CT-2005-19669 "ESTRELA".

Appendix A

Detailed Notes on Correlation: Step by Step

In this example we have an observation with observation ID “example”. In one directory we have the VEX file and the antenna log files.

```
# ls
example.skd
exampleef.log
examplema.log
examplemc.log
examplewz.log
```

We will generate all input files in this directory. The output will also be stored here.

A.1 Preparing the Input Files

Process the VEX file to create the `.input` file

```
# vex2config.pl example
# ls -rt
...
example.input
```

or

```
# vex2difx example
# ls -rt
...
example.input
example.calc
```

If vex2difx is used you will probably want to delete the `.calc` file.

Edit the `.input` file by hand

– Set the paths to the input files and to the output data

– Set the number of channels and integration time for each source

For this every configuration in the configurations table has to be set (i.e. one for each source)

```
INT TIME (SEC):     4
NUM CHANNELS:       32
```

– Remove any unused antennas

Alternatively this can be changed in the textscvex file

– Add paths to the broadband data

These will look something like this:

```
# DATA TABLE #######
D/STREAM 0 FILES:   5
FILE 0/0:           /data/SP-tmp/jm/corr1_ef_no0001
FILE 0/1:           /data/SP-tmp/jm/corr1_ef_no0002
FILE 0/2:           /data/SP-tmp/jm/corr1_ef_no0003
FILE 0/3:           /data/SP-tmp/jm/corr1_ef_no0004
FILE 0/4:           /data/SP-tmp/jm/corr1_ef_no0005
D/STREAM 1 FILES:   5
FILE 1/0:           /data/SP-tmp/jm/corr1_mc_no0001
FILE 1/1:           /data/SP-tmp/jm/corr1_mc_no0002
FILE 1/2:           /data/SP-tmp/jm/corr1_mc_no0003
FILE 1/3:           /data/SP-tmp/jm/corr1_mc_no0004
D/STREAM 2 FILES:   5
FILE 2/0:           /data/SP-tmp/jm/corr1_ma_no0001
FILE 2/1:           /data/SP-tmp/jm/corr1_ma_no0002
FILE 2/2:           /data/SP-tmp/jm/corr1_ma_no0003
FILE 2/3:           /data/SP-tmp/jm/corr1_ma_no0004
D/STREAM 3 FILES:   5
FILE 3/0:           /data/SP-tmp/jm/corr1_wz_no0001
FILE 3/1:           /data/SP-tmp/jm/corr1_wz_no0002
FILE 3/2:           /data/SP-tmp/jm/corr1_wz_no0003
FILE 3/3:           /data/SP-tmp/jm/corr1_wz_no0004
```

Process the VEX file to create the `.calc` file

```
# vex2calc.py example
# ls -rt
```

```
...
example.calc
```

Edit the `.calc` file by hand

– Remove any unused antennas and (optionally) any unused scans

`calcif` will run more quickly with a limited number of scans, however the correlator can work perfectly fine if the `.uvw` and `.delay` files cover a greater range than that which is being correlated

Process the log files to add the clock data to the `.input` file

```
# log2input.py example "exampleef.log examplema.log examplemc.log examplewz.log"
```

Process the log files to produce the ANTAB file and/or `difx2fits` input files.

This step is not entirely automatic. We essentially need to tabulate the Tsys data.

Start with an existing ANTAB file which contains the gain curves for the antennas you want. Run `log2tsys.py` once in order to determine the names of the subbands:

```
# log2tsys.py exampleef.log
```

There will be some columns which are the averages over several subbands and others which are the Tsys values for the individual subbands. Specify these in the correct order and `log2tsys.py` will print them correctly

```
# log2tsys.py exampleef.log "u1 u2 u3 u4"
```

Optionally the observer comments can be extracted from the log files and added to the ANTAB file using `log2comments.py`

Create the `.machine` and `.thread` file

```
# machinegen.py example
# ls -rt
example.machine
example.thread
```

A.2 Geometric Modelling

Generate the `.uvw`, `.delay` and `.rate` files.

```
# calcif.py example
# ls -rt
...
example.delay
example.rate
example.uvw
```

A.3 Correlation

Run the correlator

This is as simple as running

```
# mpifxcorr.py example
```

However we recommend running

```
# screen -r
# screen
# mpifxcorr.py example
```

Gnu screen runs a 'shell within a shell' such that if the machine where the operator is sitting crashes (or the connection breaks) the correlator continues to function. This can be done deliberately by typing `ctrl-a d` (d for detach). The screen can then be 'reattached' by typing `screen -r`.

Care *must* be taken to ensure every screen session is exited once the correlation is finished. Exit the shell in the usual manner making sure that you see the following line:

```
[screen terminating]
```

It is good practice to run `screen -r` before each new screen session to ensure that old sessions aren't still running.

The log file will be generated in real time. You may want to keep an eye on things by typing

```
# grep error log
```

once in a while.

A.4 Conversion and Post-processing

Next we run `difx2fits.py`

```
#ls -rt
example.skd
exampleef.log
examplema.log
```

```
examplemc.log
examplewz.log
example.input
example.calc
EXAMPLE.ANTAB
example.machine
example.delay
example.rate
example.uvw
log.2
log.1
log
example.difx/
# difx2fits.py example
#ls -rt
...
example.difx/
EXAMPLE.FITS
```

Copy the fits file to an appropriate directory

Read the data into AIPS using FITLD or run the EVN pipeline

Appendix B

Instructions on Installation

Instructions on installation can be found in other sources.

B.1 DiFX and vex2config

DiFX and vex2config can be installed following the instructions on Adam Deller's website [17]. DiFX can be taken from the website or the SVN archive, but please note that we are *not* currently using the NRAO-DiFX version of the correlator.

B.2 NRAO-DiFX

These tools are can be installed checking out a version of the SVN archive from around 1 May 2008. Comprehensive installation instructions can be found in the README for each of the tools.

B.3 IRA-DiFX

These tools can also be found in the SVN archive. They are installed following the instructions in INSTALL in the main directory.

Appendix C

Example pydifx script

```
#! /usr/local/bin/ParselTongue
"""
This pydifx/ParselTongue script carries out the correlation scan by scan,
then assembles the resulting visibilities in a single FITS file.

The .input file and .calc file have already been generated.
"""
from AIPS import AIPS
from AIPSData import AIPSUVData
from AIPSTask import AIPSTask
AIPS.userno = 142

from os import rename

import difxlog as log
from pydifx import DifxJob

rootname = 'IRACORR1'
root = '/data/SP-1/IRACORR1/080305a/' + rootname
fitsname = rootname + '.FITS'
mk5root = '/data/SP-1/IRACORR1/4stations/corr1_'

# Create Correlator object
c = DifxJob(rootname)

# Generate machine and thread files
c.machinegen()
nscans = int(c.get_calc('NUM SCANS'))

c.set_input('OUTPUT FILENAME', root + '.difx')
```

```
for i in range(1, nscans):
    c.set_input('FILE 0/0', mk5root + 'ef_no' + '%04d' % i)
    c.set_input('FILE 1/0', mk5root + 'mc_no' + '%04d' % i)
    c.set_input('FILE 2/0', mk5root + 'ma_no' + '%04d' % i)
    c.set_input('FILE 3/0', mk5root + 'wz_no' + '%04d' % i)
    exectime = float(c.get_calc('SCAN ' + str(i - 1) + ' POINTS')) *\
               float(c.get_calc('INCREMENT (SECS)'))
    c.set_input('EXECUTE TIME (SEC)', str(exectime))
    startsecs = float(c.get_calc('START HOUR')) * 3600 +\
                float(c.get_calc('START MINUTE')) * 60 +\
                float(c.get_calc('START SECOND')) +\
                float(c.get_calc('SCAN ' + str(i - 1) + ' START PT'))
    c.set_input('START SECONDS', str(startsecs))

    #run the correlator
    if i > 1:
        c.killdifx()
    log.info('Starting Correlator')
    c.go()
    log.info('Correlator Finished')

    #convert to fits
    c.difx2fits(rootname, fitsname, delete = True)

    #add to aips
    data = AIPSUVData(c.get_calc('OBSCODE'), 'UVDIFX', 1, 1)
    fitld = AIPSTask('fitld')
    fitld.infile = fitsname
    fitld.outdata = data
    fitld.doconcat = 1
    fitld.go()

    #rename fits file
    rename(fitsname, root + '%04d.FITS' % i)

# output aips fits file
fittp = AIPSTask('fittp')
fittp.indata = data
fittp.outfile = rootname + 'ALL.FITS'
fittp.format = 3
fittp.go()
```

Bibliography

[1] (2008) The European VLBI Network. [Online]:
http://www.evlbi.org/

[2] R. C. Walker, "Very Long Baseline Interferometry," in *Synthesis Imaging in Radio Astronomy II*, ser. Astronomical Society of the Pacific Conference Series, G. B. Taylor, C. L. Carilli, and R. A. Perley, Eds., vol. 180, 1999, pp. 433–+.

[3] A. R. Thompson, J. M. Moran, and G. W. Swenson, *Interferometry and synthesis in radio astronomy*. New York, Wiley-Interscience, 1986, 554 p., 1986.

[4] W. Alef, "A Review of VLBI Instrumentation," in *European VLBI Network on New Developments in VLBI Science and Technology*, R. Bachiller, F. Colomer, J.-F. Desmurs, and P. de Vicente, Eds., 2004, pp. 237–244.

[5] (2008) The 9th European VLBI Network Symposium. Presentations. [Online]:
http://www.ira.inaf.it/meetings/evn9/presentations/

[6] (2003) Mark5 VLBI Data System. [Online]:
http://www.haystack.mit.edu/tech/vlbi/mark5/index.html

[7] (2002) VEX File Definition/Example. [Online]:
http://www.haystack.mit.edu/tech/vlbi/mark5/vex.html

[8] (2001) The Earth Orientation Parameters. [Online]:
http://www.iers.org/MainDisp.csl?pid=95-87

[9] O. J. Sovers, J. L. Fanselow, and C. S. Jacobs, "Astrometry and geodesy with radio interferometry: experiments, models, results," *Reviews of Modern Physics*, vol. 70, pp. 1393–1454, Oct. 1998.

[10] (2007) Field System Index. [Online]:
http://www.naic.edu/~astro/aovlbi/fsdoc/fsindex-full.html

[11] H. J. V. Langevelde, “GPS Clocks in the EVN; Toward Blind Correlation,” JIVE, Tech. Rep., 1996. [Online]: www.evlbi.org/tog/gps/gps201.ps

[12] A. T. Deller, S. J. Tingay, M. Bailes, and C. West, “DiFX: A Software Correlator for Very Long Baseline Interferometry Using Multiprocessor Computing Environments,” *pasp*, vol. 119, pp. 318–336, Mar. 2007.

[13] E. W. Griesen, “The FITS Interferometry Data Interchange Convention,” AM113.pdf, NRAO, Tech. Rep., 2008. [Online]: http://www.aoc.nrao.edu/~egreisen/

[14] (2008) SCARIe. [Online]: http://www.jive.nl/dokuwiki/doku.php/scarie:scarie

[15] (2007) Mark-5 VLBI Analysis Software Calc/Solve. [Online]: http://gemini.gsfc.nasa.gov/solve/

[16] W. Brisken, “A Guide to Software Correlation Using NRAO-DiFX Version 1.0,” NRAO-DiFX-UserGuide.pdf, NRAO, Tech. Rep., 2008. [Online]: http://www.aoc.nrao.edu/~wbrisken/NRAO-DiFX-1.0/

[17] (2007) The DiFX homepage. [Online]: http://astronomy.swin.edu.au/~adeller/software/difx/

[18] (1995) MPI: A Message-Passing Interface Standard. [Online]: http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html

[19] (2008) MPICH-A Portable Implementation of MPI. [Online]: http://www-unix.mcs.anl.gov/mpi/mpich1/

[20] (2008) Intel® Integrated Performance Primitives 5.3. [Online]: http://www.intel.com/cd/software/products/asmo-na/eng/302910.htm

[21] (2008) AMD Performance Libraries. [Online]: http://developer.amd.com/cpu/Libraries/Pages/default.aspx

[22] (2008) Python Logging. [Online]: http://www.python.org/doc/2.5.2/lib/module-logging.html

[23] (2008) The Parseltongue Wiki. [Online]: http://www.jive.nl/dokuwiki/doku.php/parseltongue:parseltongue

[24] (2008) EVN Pipeline Feedback for User Experiments. [Online]: http://www.evlbi.org/pipeline/user_expts.html

[25] W. Brisken, “A Guide to Software Correlation Using NRAO-DiFX Version 1.1,” NRAO-DiFX-1.1-UserGuide.pdf, NRAO, Tech. Rep., 2008. [Online]:
http://www.aoc.nrao.edu/~wbrisken/NRAO-DiFX-1.1/

[26] (2007) CASA. [Online]:
http://casa.nrao.edu/

[27] (2008) Open MPI: Open Source High Performance Computing. [Online]:
http://www.open-mpi.org/