

# Atacama Large Millimeter Array

ALMA-SW-0NNN

Revision:0.1

2003-03-13

*Software Tests*

Robert Lucas

## IRAM Aips++ Test: Phase III

*Software Tests*

R. Lucas, D. Broguière, K. Golap, R. Rusk

<b>Keywords:</b> Software Test	
Author Signature: Robert Lucas	Date: 2003-03-13
Approved by: B. Glendenning, K.-Y. Morita, G. Raffi	Signature:
Institute: NRAO, NRO, ESO	Date:
Released by:	Signature:
Institute:	Date:

*Change Record*

Revision	Date	Author	Section/ Page affected	Remarks
0.1	2001-03-13	Robert Lucas	All	Initial Version

\$Id: PhaseIIIDraft.tex,v 1.5 2003/03/15 00:33:24 lucas Exp lucas \$

## Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Data Set description</b>	<b>4</b>
<b>3</b>	<b>Gildas processing</b>	<b>4</b>
<b>4</b>	<b>Data Set Processing using Aips++</b>	<b>5</b>
<b>5</b>	<b>Preliminary trends and conclusions</b>	<b>5</b>

## 1 Introduction

The main objective of Phase III of this test is to capitalize on the effort put into performing Phase I, in order to explore computing performance issues raised by millimeter-wave oriented data reduction of ALMA-sized data sets in Aips++. It represents a first step in the monitoring of AIPS++ performance on realistic ALMA data sets that the SSR plans to define in the near future and are referred to in the off-line data processing requirements document (ALMA Software memo 18).

The data reduction functionalities that have been the subject of phase I and II are typical of present-day millimeter wave interferometry (low signal to noise calibration, high relative importance of spectral line imaging); it thus appeared worthwhile to start studying Aips++ performance using those low noise algorithms. Also the opportunity is offered of a direct comparison of end-to-end data reduction performance (from raw data fits files to images) with an existing operational package, as the performance of the software in current use at IRAM can be used as a starting point reference for the evaluation of further progress in Aips++ development. The results of this test can (and already have been) used as a guide for future development, by identifying the areas where work is most needed.

**The present text should be considered as a progress report.** Many of the results presented here are **very preliminary**, and deserve a lot of further detailed analysis. In certain cases the runs did not terminate properly, so the numbers could be wrong. In many cases we have not checked the integrity of the results and even looked at the final images.

Section 2 describes how the data set was built. Section 3 presents the results of processing with Gildas. Section 4 gives the first results using Aips++, and preliminary conclusions are given in Section 5.

## 2 Data Set description

The data set was obtained in the following way:

1. Select an approximately two hour period of one day of the Phase I data set: (25-mar-1997, scans 7306 to 7429).
2. Create from this data a N-antenna Plateau de Bure data set:
  - Header data for each antenna is cloned from that of antenna N (modulo 5) of the original header data (5 was the actual number of available antennas at Plateau de Bure in 1997).
  - All visibilities are replaced by point source visibilities, using the source flux for calibrators, and 0.07 Jy for the target source (GGTau). Random thermal noise is added corresponding to the channel width and system temperatures.
  - Antenna positions are replaced with random positions (with a dispersion of 50m, max baseline 300m), and a minimum separation of 24m. U and V are re-computed accordingly.
3. Convert the data set into ALMATI-FITS.

Four data sets were prepared with  $N = 8, 16, 32, 64$ . The data set sizes are 139, 465, 1774, and 6979 Megabytes (fits files).

The actual integration time of the original data set was 106 min. The simulated visibility data rate is thus about 1.1 MB/s (12.5% of the specified average visibility rate, if we take into account that those fits files contain 8-byte visibilities, not 4-byte as specified for ALMA).

## 3 Gildas processing

As a validation test of the data set, and also to get a feeling of the difficulty of Phase III, we processed the data in Gildas using an automatic procedure. The data set is simple enough that the reduction is fully automatic (input fluxes are given, no phase or amplitude instrumental or atmospheric errors). The starting point is the ALMATI-FITS format, the end point is displayed images (Postscript).

The changes in CLIC for this test were:

- Re-dimension it to 8, 16, 32, 64 antennas.
- Change a few formats that would accommodate only one-digit antenna numbers.

The processing was made on a AMD 1532 MHz PC with 768 MB of memory and 256KB of cache memory. All data was on a local disk (60 GB, DMA enabled). Use of a PC with 1024 MB of memory did not appreciably improve the results.

Table 1 contain the timing results (in elapsed time).

The times are split between the various steps:

1. Data filling from ALMATI-Fits (program `tifits`)
2. Calibration in CLIC:
  - (a) Create a header file and select scan numbers
  - (b) RF Bandpass calibration at 3mm and 1mm
  - (c) Phase calibration at both frequencies, including phase transfer from 3mm to 1mm.
  - (d) Flux calibration at 3mm and 1mm
  - (e) Amplitude calibration at 3mm and 1mm
3. Create visibility tables using CLIC (3mm continuum, 3mm line, 1mm continuum, 1mm line).
4. Total imaging: create raw images with GRAPHIC task `UV_MAP`. Mapping parameters are those of phase I (256 by 256 images, 1 channel for line, 48 channels for continuum, uniform/robust weighting); clean the images with GRAPHIC task `CLEAN` (Clark method).
5. Total processing time.

As an indication the last three rows of Table 1 give:

1. The data processing rate (obtained by dividing the fits file sizes by the total times);
2. The current SSR specified rate (doubled to take into account that the ALMA data should be recorded as 4-byte visibilities, while in the FITS files used here one visibility is stored as 8 bytes). This is scaled down by the square of the antennas for 8, 16, 32 antennas data sets.
3. The ratio of the above.

Some results were also obtained on a larger memory machine: `bernouilli`, with 8 GB memory, 8 PIII processors at 700 MHz, with 2MB cache each. The 64 antenna test could not be run. The results are shown on Table 2

## 4 Data Set Processing using Aips++

The data was recently processed on the same computer with Aips++ stable release (AIPS++ version: 1.8 build #499). It uses the `almati2ms` and `iramcalibrator` tools.

The results for the same steps are given in Table 3. Note that the init step (see above) corresponds to different initialization operations in the two packages.

In Table 4 we show the ratio of processing times between the two packages on `iralx7`.

As it clearly appeared during processing that intense paging activity occurred during the time consuming steps, the tests were run on PC with a large memory (`bernouilli`, 8 GB memory, 8 PIII processors at 700 MHz, with 2MB cache each). The results for Aips++ are in Table 6.

## 5 Preliminary trends and conclusions

The graphs in figures 1 and 2 show the relative time spent in each main step for each package, on the `iralx7` computer (768 MBytes). Fig 3 shows the ratio of processing times for each step. Similar plots for data processing on `bernouilli`, which has a slower CPU but enough memory to prevent paging losses, are on figs 4 to 6.

What should be noted is:

Table 1: Gildas result on iralx7: AMD processor at 1500 MHz, 768 MB memory, 256KB cache memory. The ‘spec’ row is the SSR specified rate (however doubled as we handled 64-bit visibilities, not 32 bit as specified). The last row is the ratio of the achieved processing rate to the specified one.

nant	8	16	32	64
Program	Clic08	Clic16	Clic32	Clic64
filler	18.8	86.7	214.5	815.7
init	1.4	20.9	65.6	226.8
phcor	3.8	24.3	108.8	405.5
rf passband	4.9	35.8	309.7	2267.1
phase	1.6	10.6	107.4	1301.5
flux	3.1	7	72.2	832.3
ampli	1.5	7.1	79.7	922.4
total cal	16.3	105.7	743.4	5955.6
table	6.5	39	192.9	562.3
imaging	24.5	25.5	44.5	190.7
total	66.1	256.9	1195.3	7524.3
fits size (MB)	139	465	1774	6979
rate (MB/s)	2.1	1.81	1.48	0.93
spec (MB/s)	0.125	0.5	2	8
ratio	16.8	3.62	0.74	0.12

Table 2: Gildas results on bernouilli: 8 GB memory, 8 PIII processors at 700 MHz, with 2MB cache each. The 64 antenna column (in red) is estimated from iralx7 numbers, scaled by the bernouilli/iralx7 performance ratio estimated from the 32 antenna results.

nant	8	16	32	64
Program	Clic08	Clic16	Clic32	Clic64
filler	19.6	90.2	389.3	1480.4
init	4	12.8	56.7	196.0
phcor	7.4	31.2	156.3	582.5
rf passband	10.9	81.5	599.1	4385.6
phase	3	18.1	239.4	1693.4
flux	4.3	13.5	146.9	1693.4
ampli	3.2	16.9	162.9	1885.3
total cal	32.8	174	1361.3	11643.9
table	13.8	42.6	228.2	665.2
imaging	38.5	38.7	67.9	291.0
total	104.7	345.5	2046.7	14080.5
fits size (MB)	139	465	1774	6979
rate (MB/s)	1.33	1.35	0.87	0.50
spec (MB/s)	0.125	0.5	2	8
ratio	10.64	2.7	0.43	0.06

Table 3: Aips++ result on iralx7: AMD processor at 1500 MHz, 768 MB memory, 256KB cache memory.

nant	8	16	32	64
Program	Aips++	Aips++	Aips++	Aips++
filler	114.6	346.9	1108.7	4016.1
init	26.3	132.8	622.9	2688.4
phcor	22.6	78.7	293.8	1280.1
rf passband	28.5	110.3	692.9	5078.6
phase	24	84.1	546.3	13846.3
flux	26.8	64.8	197	871.6
ampli	21	74.4	495.9	13558.8
total cal	149.2	545.1	2848.8	37323.8
table	61.2	107.4	489	2486.6
imaging	160.7	181	384.4	1151.8
total	485.7	1180.4	4833.1	44978.3
fits size (MB)	139	465	1774	6979
rate (MB/s)	0.29	0.39	0.37	0.16
spec (MB/s)	0.125	0.5	2	8
ratio	2.32	0.78	0.19	0.02

Table 4: Ratio of processing times (Gildas/Aips++) on iralx7: AMD processor at 1500 MHz, 768 MB memory, 256KB cache memory.

nant	8	16	32	64
filler	6.096	4.001	5.169	4.924
init	18.786	6.354	9.495	11.854
phcor	5.947	3.239	2.7	3.157
rf passband	5.816	3.081	2.237	2.24
phase	15	7.934	5.087	10.639
flux	8.645	9.257	2.729	1.047
ampli	14	10.479	6.222	14.699
total cal	9.153	5.157	3.832	6.267
table	9.415	2.754	2.535	4.422
imaging	6.559	7.098	8.638	6.04
total	7.348	4.595	4.043	5.978
rate ratio:	0.138	0.215	0.25	0.172

Table 5: Aips++ result on bernouilli: 8 GB memory, 8 PIII processors at 700 MHz, with 2MB cache each. Note that the imaging step got stuck for 32 and 64 antennas. The red numbers are extrapolated guesses.

nant	8	16	32	64
Program	Aips++	Aips++	Aips++	Aips++
filler	273.1	774.1	2765.7	10982.7
init	42.3	134.1	568.3	2140.6
phcor	56.8	207.4	836.3	3394.9
rf passband	31.2	113.9	477.8	2437.1
phase	15.9	54.1	216	1236.9
flux	40.3	117.8	422.7	2019.9
ampli	39.3	132.3	495.9	3352.2
total cal	225.8	759.6	3017	14581.6
table	70.3	174	832.4	4848.7
imaging	186.2	249.4	529.7	1587.2
total	755.4	1957.1	4833.1	31564.8
fits size (MB)	139	465	1774	6979
rate (MB/s)	0.18	0.24	0.37	0.22
spec (MB/s)	0.125	0.5	2	8
ratio	1.44	0.48	0.18	0.03

Table 6: Ratio of processing times (Aips++/Gildas) on bernouilli: 8 GB memory, 8 PIII processors at 700 MHz, with 2MB cache each.

nant	8	16	32	64
filler	13.93	8.58	7.10	13.46
init	10.57	10.47	10.02	9.438
phcor	7.67	6.64	5.35	8.372
rf passband	2.86	1.39	0.79	1.114
phase	5.3	2.98	0.90	1.313
flux	9.37	8.72	2.87	2.427
ampli	12.28	7.82	3.04	4.735
total cal	6.88	4.36	2.21	2.75
table	5.09	4.08	3.64	8.623
imaging	4.83	6.44	5.66	6.04
total	7.21	5.66	2.36	4.59
rate ratio:	0.13	0.17	0.42	0.21

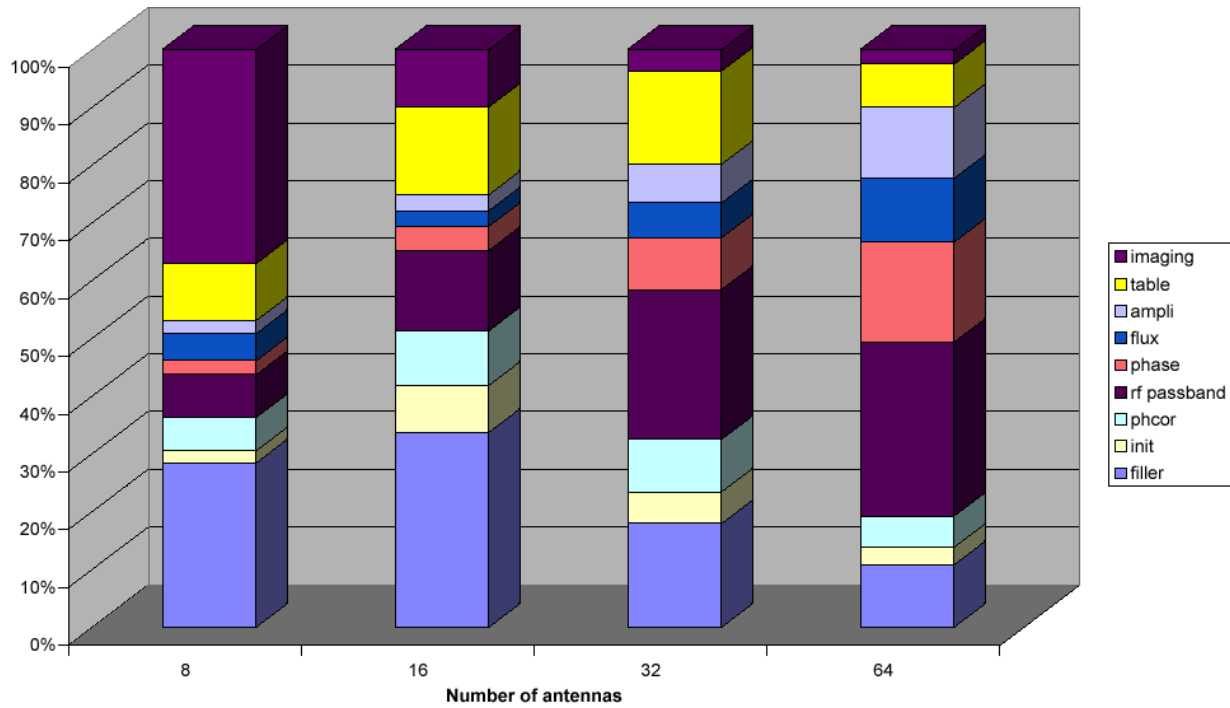


Figure 1: Gildas processing relative processing times on iralx7

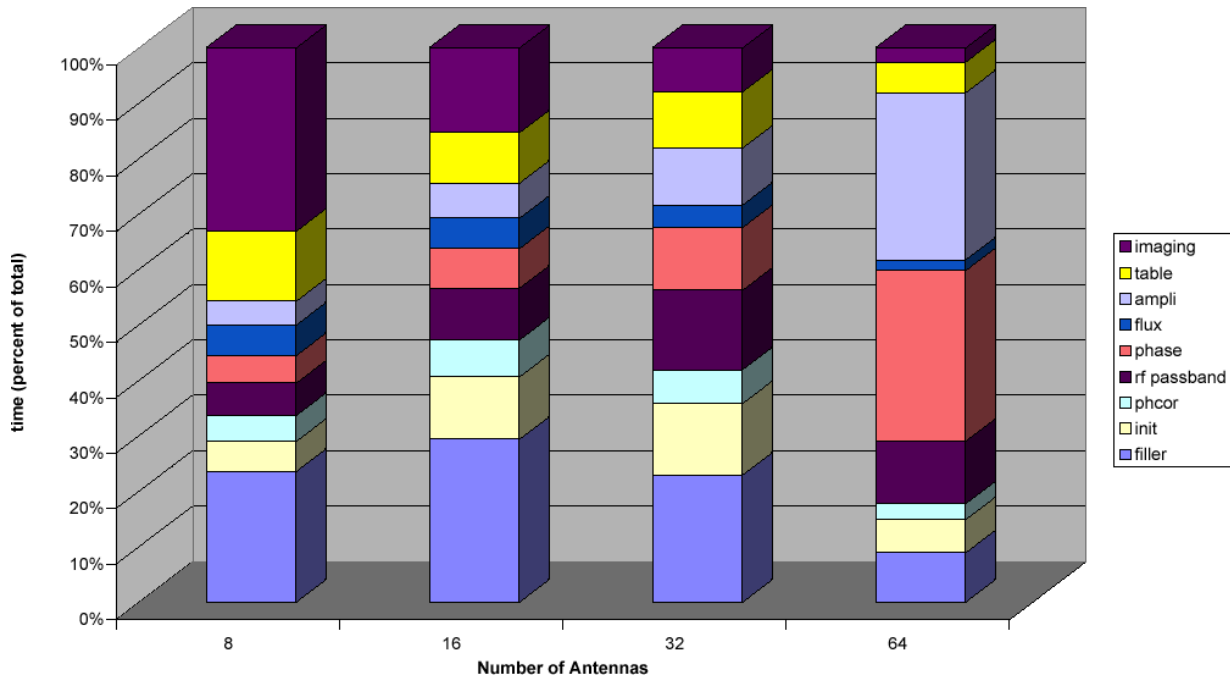


Figure 2: Aips++ processing relative processing times on iralx7

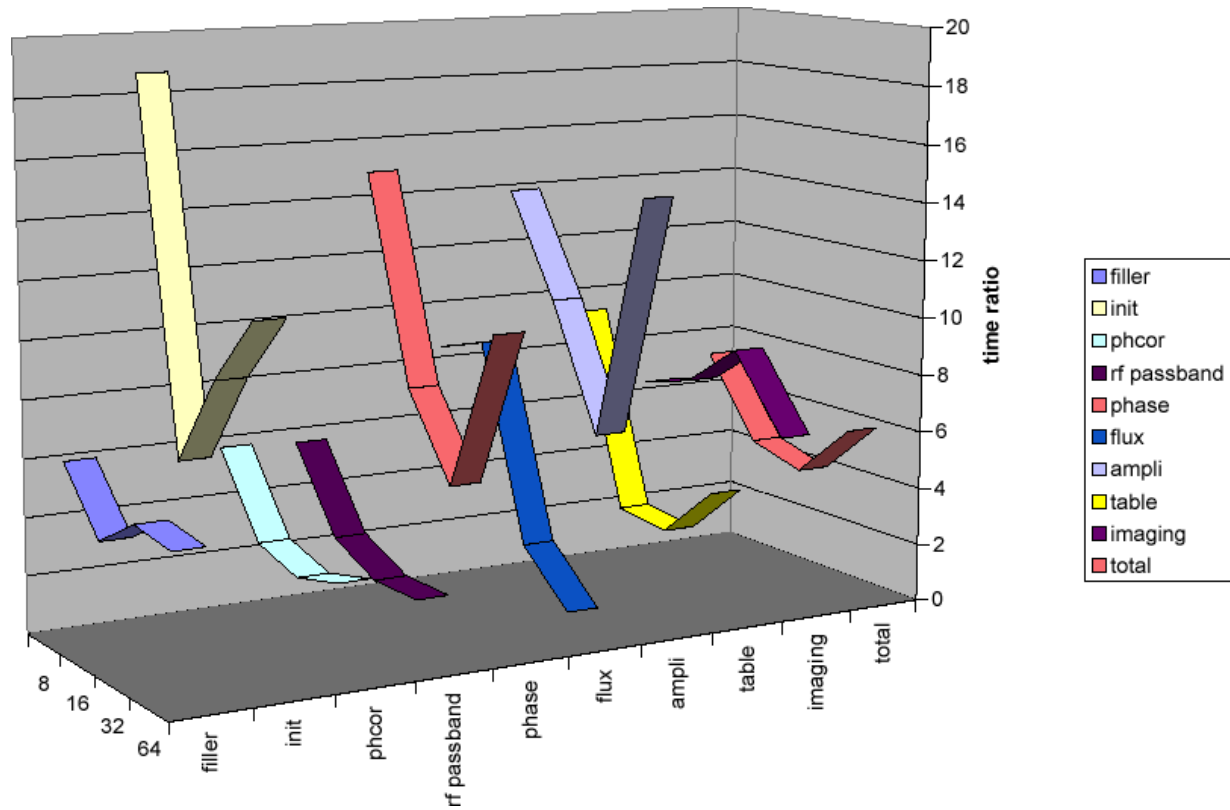


Figure 3: Aips++/Gildas processing time ratios on iralx7

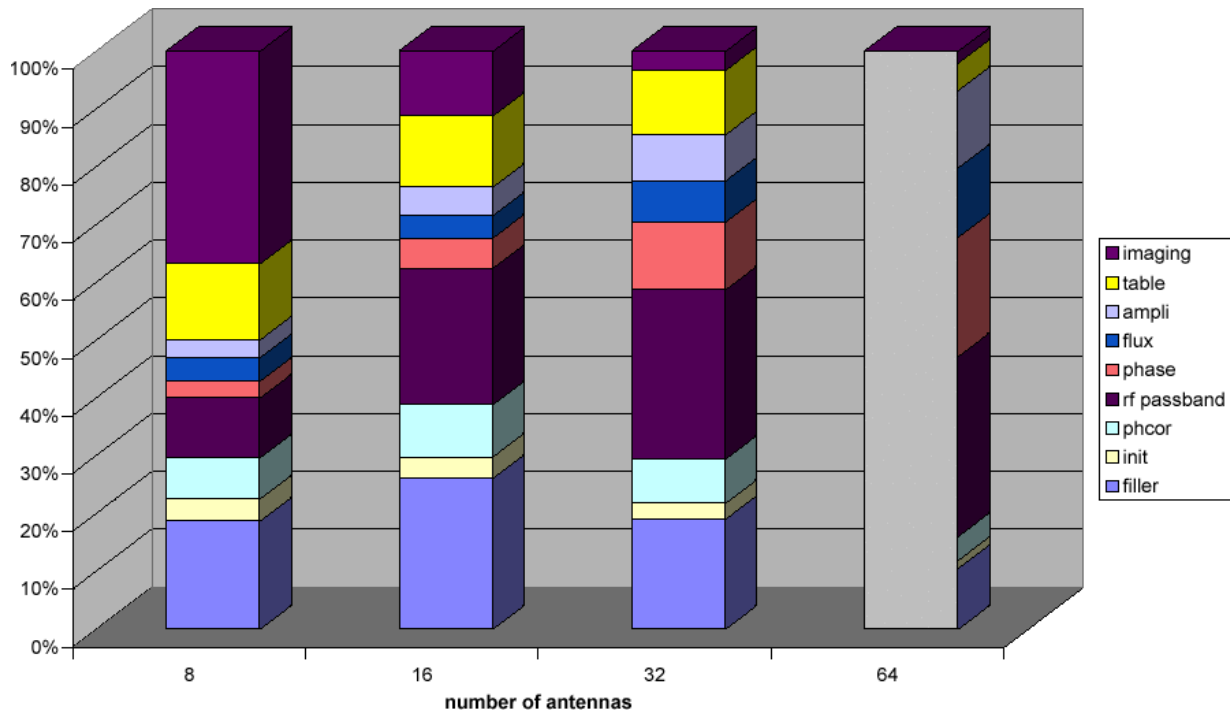


Figure 4: Gildas processing relative processing times on bernouilli

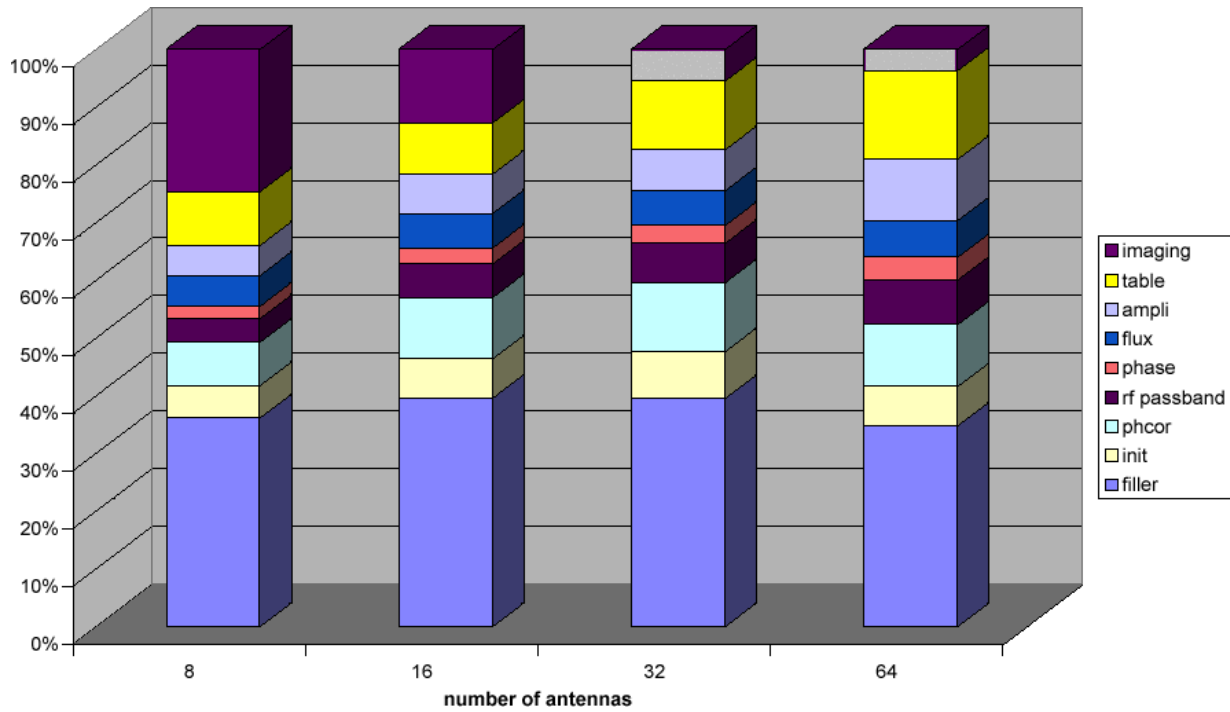


Figure 5: Aips++ processing relative processing times on bernoulli

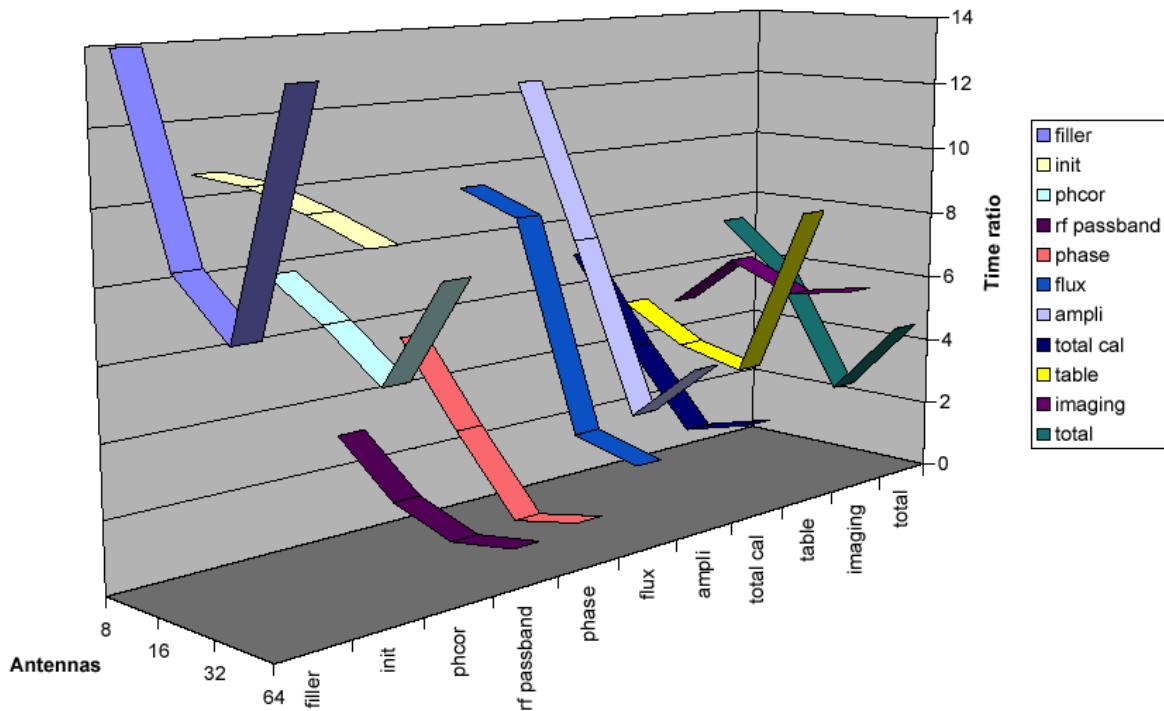


Figure 6: Aips++/Gildas processing time ratios on bernoulli

- The total processing rates with present-day desktop equipment are not far from what will be needed ALMA gets 8 – 16 antennas. We will have to count on hardware progress (Moore’s law factor is about 100 from 2002 to 2012) to process 64 antenna data (this is not a surprise, and the plan is also to have Linux clusters for the pipelines). Nevertheless we all know that the specified data rate is very likely to increase.
- As can be seen the data processing needs (for this example) are largely dominated by calibration, for both packages.
- Aips++ is slower than Gildas on the average by a factor of a few, for all four data sets. At present to obtain the best results (2.4 ratio), Aips++ needs much more memory (several Gbytes). With less than 1 Gbyte ratios are of order 4 – 5
- The basic core algorithms for the most CPU heavy operations (rf passband, phase, amplitude calibrations) are the same for both packages (essentially two Fortran subroutines, polyant and splinant, that do antenna based polynomial and spline interpolation for amplitudes and phases). The differences in performance most probably result from the different ways the data is handled in the two packages. Obviously these steps are greatly improved in Aips++ by using a larger core memory; this is not the same for Gildas; CLIC being written entirely in Fortran is using very few memory transfers. The times for the “PHASE” steps with the large memory indicate that this is probably not the case for Aips++.
- Other steps are doing data organisation rather than numerical calculations: FILLER, INIT, TABLE. The relative times between the two packages seem rather independent of the memory size. As a consequence on bernouilli the data filler uses more than 35% of the total processing time.
- “PHCOR” and “FLUX” are, in Aips++, implemented using glish. It is thus expected that they should be much less efficient than similar ones implemented in C++. However the relative ratio gildas/aips++ for these steps are not that different from the FILLER, INIT, TABLE steps. This is probably worth looking into.

Directions of research in the near future:

- What is the proportion of the time spent in applying the calibrations in Aips++ ? In CLIC this is only done on-the-fly when the data is needed: as an input for a further calibration step and when building the final visibility tables. The latest version of iramcalibrator (used on bernouilli) in Aips++ only applies the calibration in the ”AMPLITUDE” step. It would be desirable to measure this separately in both packages (in Gildas this is done only in the “TABLE” step).
- How much of the difference in the filler and other data reorganization steps is due to:
  - The way the data is organized (measurement set data model)
  - The way the data is stored (data managers, tiling parameters)
  - The way the data is actually used (kept in memory as much as possible in the same place).