Benchmarking in AIPS++

Sanjay Bhatnagar

NRAO, Socorro

Approach to benchmarking

- ***Develop easy-to-run tools***

    Populate the AIPS++ Benchmark module

    Make same and repeatable measurements

- ***Develop conceptual understanding of the compute, I/O and other major costs of the algorithms***

- ***Reduce (eliminate) extraneous parameters***

- ***Understand performance curves (and not spot measurements) as a function of the problem size***

- ***Start from the simplest, but realistic problem size***

Work so far...

- ***Image re-gridder optimization***
  - Low level optimization (Brouw)
  - Achieved a factor of ~10 improvement. Real-time image re-gridding possible
- ***Interferometric imaging and calibration: Study of the algorithms and the associated I/O costs***
- ***Equivalent scripts for other popular packages (AIPS, Miriad)***
- ***Profiling: Very useful and in progress using standard GNU tools***
- ***Regular profiling builds of AIPS++***
- ***Dedicated computer for benchmarking***
  - Need isolation and controlled environment

Work so far...

- ***Developed benchmark.g***

  Control on algorithms, problem size, data size, Table system parameters, memory model...

- ***ALMA data size tests***
  - First cut at ALMA TI filler (Rusk)
  - Iramcalibrater compared with CLIC.

- ***Well known bottle-necks being optimized***

- ***Profiling and optimization in progress***

Table 1.1381: Table for continuum VLA 1M rows dataset benchmark. Common part of the benchmark code: 'CC-SF-VLAC-U1M-SP1-'

Benchmark code	Dataset	N_{spw}	Stokes	Wgt.	N_{pixel}	N_{chan}	N_{clean}
I-UN-512-C1-1000	vlac1M	1	I	UN	512	1	1000
IQUV-UN-512-C1-1000	vlac1M	1	IQUV	UN	512	1	1000
I-UN-1024-C1-1000	vlac1M	1	I	UN	1024	1	1000
IQUV-UN-1024-C1-1000	vlac1M	1	IQUV	UN	1024	1	1000
I-UN-2048-C1-1000	vlac1M	1	I	UN	2048	1	1000
IQUV-UN-2048-C1-1000	vlac1M	1	IQUV	UN	2048	1	1000
I-NA-512-C1-1000	vlac1M	1	I	NA	512	1	1000
IQUV-NA-512-C1-1000	vlac1M	1	IQUV	NA	512	1	1000
I-NA-1024-C1-1000	vlac1M	1	I	NA	1024	1	1000
IQUV-NA-1024-C1-1000	vlac1M	1	IQUV	NA	1024	1	1000
I-NA-2048-C1-1000	vlac1M	1	I	NA	2048	1	1000
IQUV-NA-2048-C1-1000	vlac1M	1	IQUV	NA	2048	1	1000

Table 1.1384: Table for VLA calibrator benchmark (for 'G' and 'D' Jones).

Benchmark code	Data	Compressed?	Jones	NAnt	SNR	NSolInt
CALVLAU-G-27-10-100	calvlac27s10.fits	Nope	G	27	10	100
CALVLAU-D-27-10-100	calvlac27s10.fits	Nope	D	27	10	100

Table 1.1385: Table for FITS UV read benchmark (import visibility data into AIPS++ from FITS files). Common part of the code is ' `FUV-RD-VLA`'.

Benchmark code	Dataset	Mode	N_{chans}	Compressed?	Data size
C-U125K-C1	vlac125k.fits	Continuum	1	False	125K
C-U1M-C1	vlac1m.fits	Continuum	1	False	1M
L-U125K-C64	vlal125K64Chan.fits	Line	64	False	125K

Results: General

- ***AIPS performance curves were taken as the 'baseline' (Miriad is 2x faster than AIPS though!)***

- ***AIPS++ default settings too conservative***

  - Memory/CPU usage monitoring indicated memory model too conservative for present day computers

  - Default settings can be (should be?) made computer-resource-aware

- ***Burnt-in assumptions for memory model in some critical parts of the code***

- ***Redundant (expensive) operations***

- ***Profiling ultimately dug out a few real & 'dormant bugs'***

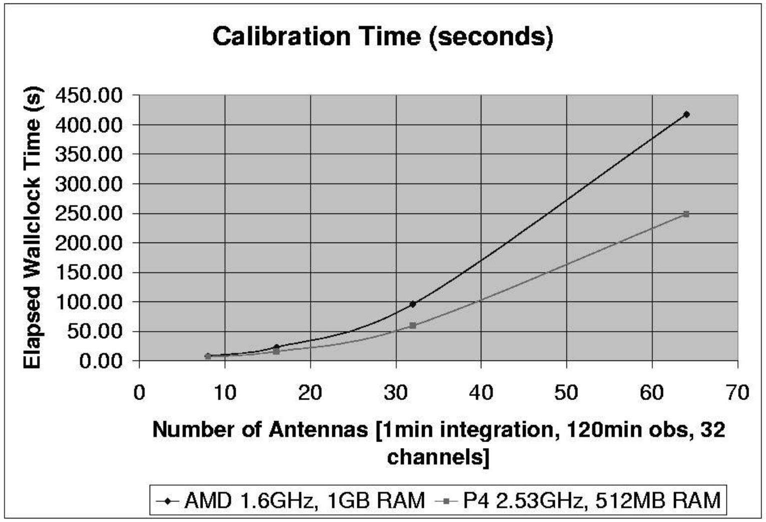
Results: Well-known mistakes

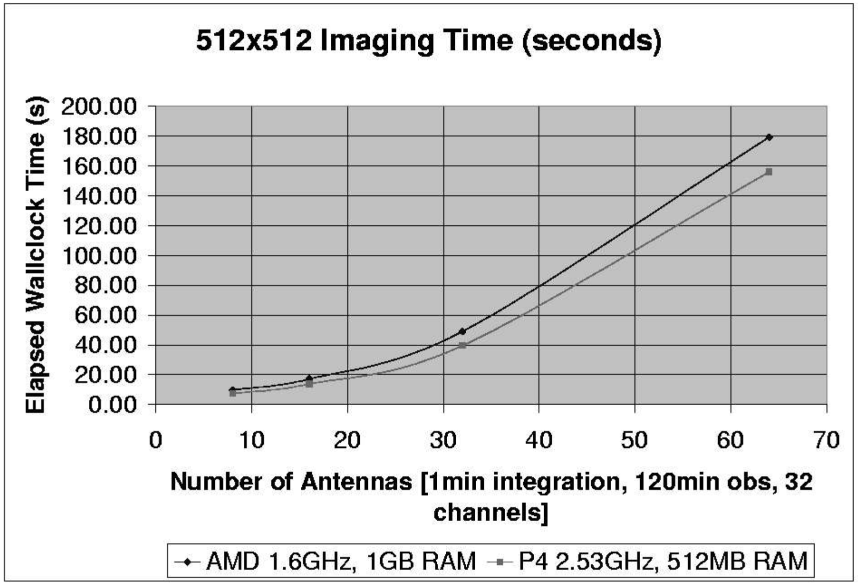
- ***Tiling* in the construction of Tables is an important parameter**
- ***Array access pattern*: can be expensive if done the wrong way**
- ***Inadvertent use of copy vs. reference semantics is expensive***
  - Results into excessive memory copies
  - Use of objcopy() should be minimized
- ***Table creation may be expensive***
  - Used by ArrayLattice in tiling mode
  - Minimize Table creation inside tight loops
- ***Strictly use only OSInfo() class for memory model***

Results: ALMA Tests

- ***Performance studied as a function of data-rate***

  (Rusk)
- ***Imaging appears to scale well***
- ***Hardware/compilers factor in as significant parameters***
- ***Understand the scaling as a function of various parameters***
- ***More work needed to understand the differences in these curves***





Results:IRAM Calibrater Tests

- ***First cut* at iramcalibrater benchmarking** (Rusk/Lucas/Golap)

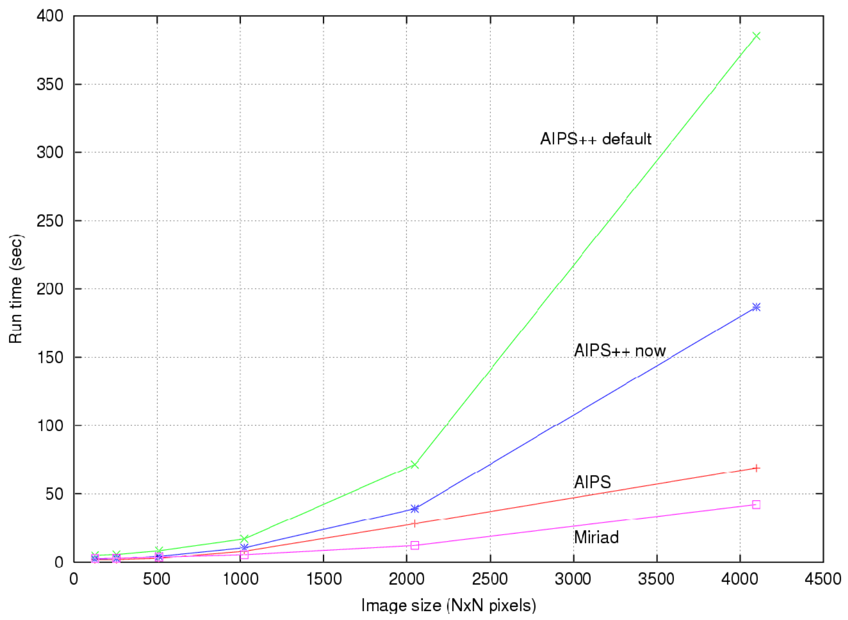
No. of antennas: 64

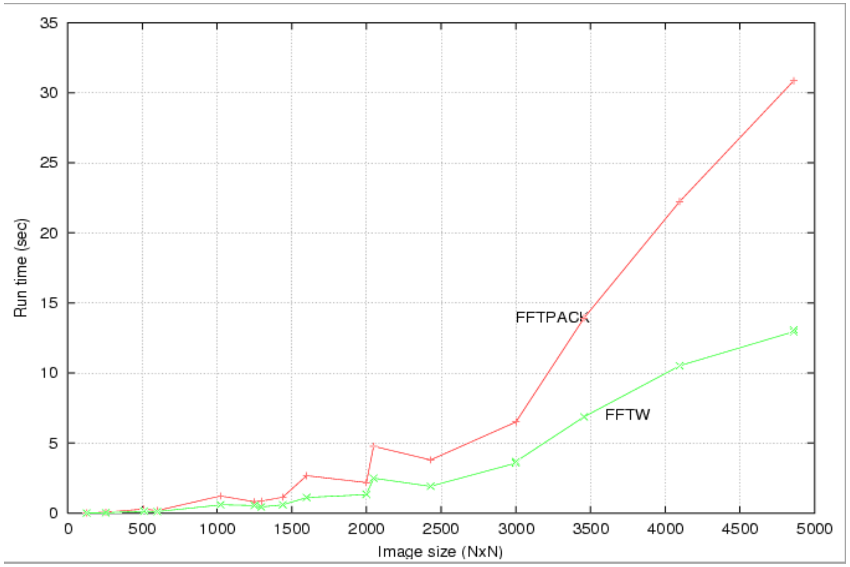
Calib. cycle	AIPS++ (sec)	AIPS++ (no plots) (sec)	CLIC (sec)
RF	2646	1818	2187
Phase	1404	804	942
Amp	2502	2076	708

- ***AIPS++ overheads insignificant***
- ***AIPS++ code not yet fully optimized***
- ***RF & Phase solvers use the same engine in AIPS++ and CLIC: runtime within 10-20% of each other***
- ***Amp: same solver, but AIPS++ includes apply()***

Results: Imager

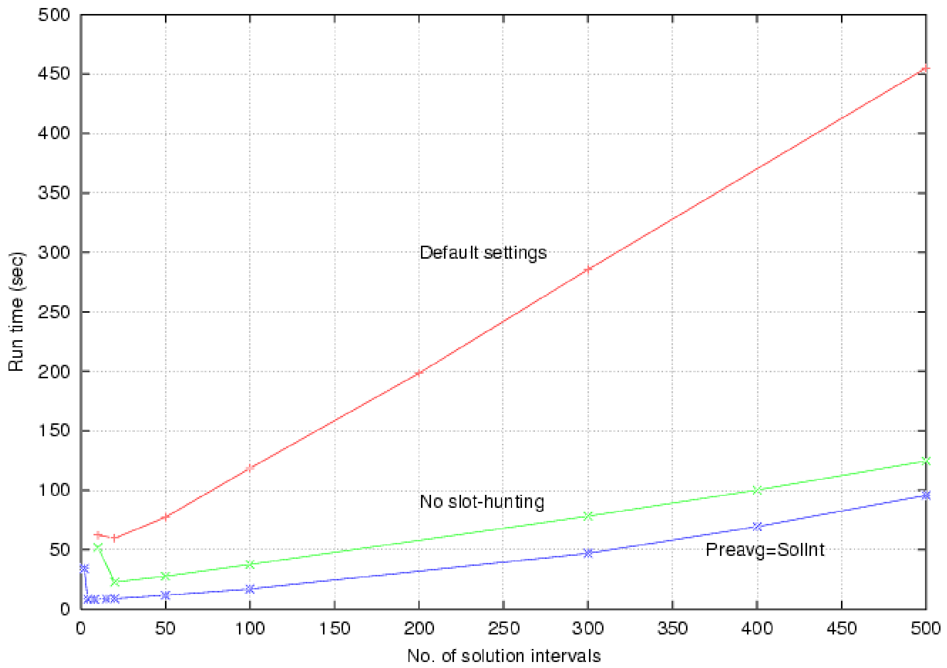
- ***Clark-CLEAN algorithm (one of the many in AIPS++)***
- ***Recognize 'minor' /subtle differences in the implementation***
  - AIPS uses Cotton-Schwab variant
- ***FFTW: optimize data locality and in-memory copies***
- ***Threaded FFT: useful on ubiquitous multi-CPU machines***





Results: Calibrater

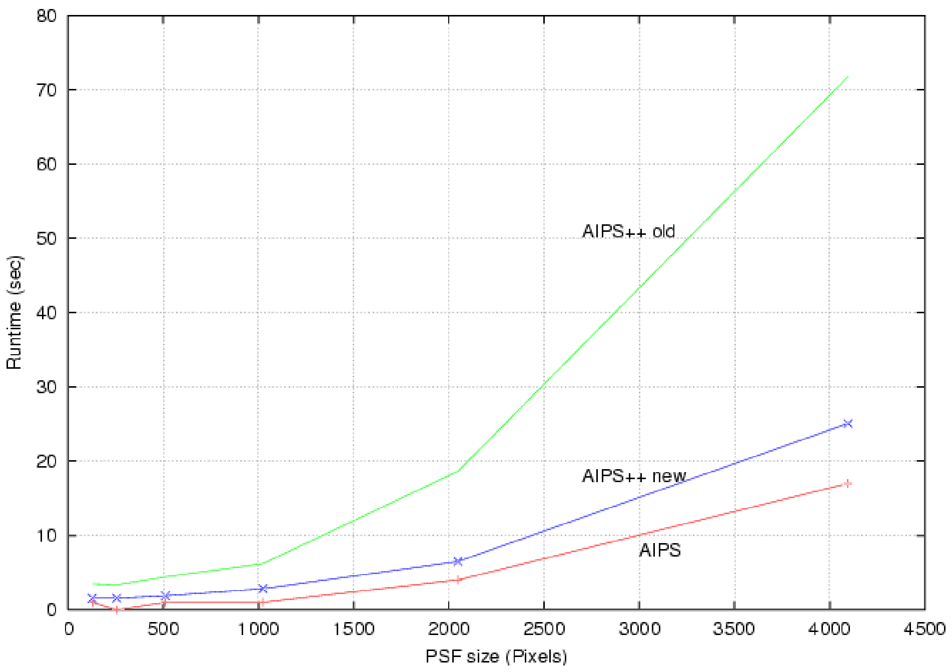
- ***Gain and band-pass calibration was not optimal (fixed)***
- ***Slot hunting algorithm was non-optimal (fixed)***
- ***Profile dominated by lower level calls***
- ***More work needed once these major bottlenecks are removed***



Gain and polarization leakage calibration comparable to AIPS till ~200 solution intervals.

Results: I/O test (not well understood)

- ***Measure I/O costs as seen by the AIPS++ 'algorithm layer'***
- ***Table system's cache-hit rate was >99%***
- ***Need more work***



Lessons learned

- ***Benchmarks are a measure in a multi-dimentional space***
  - Requires careful analysis and understanding of the software and the hardware platform
  - Various axis are not often orthogonal
- ***Benchmarking and optimization is better done closer to code development than much later.***
- ***Profiling is the single most useful tool.***
- ***No fundamental hot-spots (IMHO!)***

...continued

- ***AIPS++ framework is highly configurable***
  - Memory model is user configurable
  - Table I/O is configurable (need to bring it to the user level?)
- ***Smarter automatic (computer resource-aware) settings is an enticing possibility***
- ***Better Technical documentation***
  - Cost analysis for developers
  - Do's and Don't 's for developers