AIPS MEMO NO. 42

4 April 1986

Pseudo Array Processor Speed

W. D. Cotton

1 INTRODUCTION

This memo will report the current state of pseudo array processor code on the VAX and Convex. Recent developments of the "Q" routines on the Convex have signifigantly increased its speed; many of these developments have been moved to the VAX pseudo AP library. These improved routines, together with the use of the VAX optimizer for the VAX Q routines only have resulted in improved performance of the VAX pseudo AP.

2 RECENT DEVELOPMENTS ON THE CONVEX

Complex FFT: The AIPS benchmark trials on the Convex were run with a vectorized Fortran FFT provided by Steve Wallach of Convex. This routine depends on routines which were provided in only object form. The speed of this routine in a 1024 complex FFT is about 6 msec. Our modified version of this routine initilized the FFT tables once.

Subsequently, Convex supplied an assembler FFT which does a 1024 complex FFT in about 5 msec. This routine also depends on routines which were provided in object form only; one of these routines has the same name as one of Wallachs but apparently is different. This FFT reinitilized the tables every time the size of the FFT was changed. This initilization is quite expensive and eliminated the advantage of the extra speed. Attempts to modify the code so that the tables were initilized only when the size of the FFT was increased failed. Since we don't have source code for the initilization routines the reason for this failure could not be determined.

A new, vectorizable FFT routine was developed from a selfcontained routine provided by Fred Schwab. This routine takes about 8.5 msec for a 1024 complex FFT. The tables are initilized every time the size of the FFT is changed but since the tables are small and the initilization vectorizes, the cost of initilization is negligible. At present this routine is being used in the Convex; if we get an acceptable, faster routine from Convex it will be used in the future.

Real FFT: Our pseudo AP real-to-complex FFT did some shuffling of the data and then called the full complex FFT. This shuffling did not vectorize due to the recursive derivation of some coefficients. These coefficients are now computed once and stored in a table. This both reduces the amount of work and allows vectorization of this process.

"AP" size for FFTs: The implementation of CLEAN on the Convex uses a number of work arrays which reside in the "AP memory" common. This memory is available for other uses. The size of the "AP" for FFT was effectively quadrupled by using some of this memory. The larger AP memory size means that 512x512 FFTs can be done entirely in memory and don't need to be written to disk during the transpose. This decreases the time used by MX in a typical run by about 10%.

The larger AP size also means that the block size for disk reads in the second pass of the FFT increased by a factor of four. In large FFTs (2048x2048 and bigger) this makes a dramatic difference in the real time to do the FFT. Factors of up to 4 in the real time performance of the FFT may be realized for very large (4096x4096) FFTs. These changes affect only large FFTs and will have no effect on the timings given later.

Polar to Rectangular Conversion: Our direct Fourier transform method depends heavily on polar to rectangular conversions. This operation is very efficient on FPS array processors, but is relatively slow on the Convex and exceedingly slow on the VAX. A table lookup - series expansion technique was adopted to speed up this operation. Unfortunately, on the Convex, data type conversions are very slow and the table lookup in Fortran was slower than the sine and cosine computations. An assembler routine using a number of tricks to do the data type conversion yielded a routine about 3 times faster than the method using the Convex sine and cosine routines.

Phase Ramps: Several applications in AIPS depend on the efficient application of phase ramps to a complex vector. The technique used in the AP depends on a recursive technique and thus does not vectorize; it is also prone to truncation and roundoff problems. Vector phase ramp techniques were implemented using auxillary arrays. This results in better numerical accuracy as well as vector speed.

Gridding/Interpolation: The inverse operations of gridding and interpolation consist of a double DO loop of which only the inner loop could be vectorized as written. The length of this loop is rather short, typically 7 or 11. Some enhancement in the performance was obtained by using the gather/scatter capability of the Convex to convert the double DO loop into a single DO loop of length typically 49 or 121.

3 RECENT DEVELOPMENTS ON THE VAX

Optimization: The current Q routines appear to give the correct results when the VAX Fortran optimizer is used. However, not all code has been exercised in all modes to verify the completely correct function of the optimizer on the Q routines; optimized Q routines should be treated with some suspicion.

FFTs: Both the complex and real-to-complex Fortran FFTs from the Convex were adopted into the VAX pseudo AP. The complex FFT, when compiled with the optimizer, gives performance similar to the old assembly FFT (the current installation procedures appear to be using the old Fortran version rather than the assembly version).

Polar to Rectangular Conversion: The Fortran version of the table lookup routine from the Convex runs about 2.5 times faster than the old version and so was adopted for the VAX pseudo AP.

4 TIMINGS

The following table gives the relevant times for the PFT test AP routines. The column labeled "no. bits" gives the number of bits of accuracy for the most discrepent pixel comparing the 15JUL86 pseudo AP (PSAP) and the FPS AP. The timing tests were run on busy machines for the VAX pseudo AP tests and on otherwise empty machines for the VAX+AP and Convex tests.

Table 1
Timing of PFT AP tasks in seconds

Task	15JUL85 VAX PSAP cpu	15JUL86 VAX PSAP cpu	15JUL85 Convex cpu	15JUL85 Convex real	15JUL86 VAX AP real	no. bits
UVMAP	151	115	19	31	88	14.8(map) 10.3(beam)
APCLN	4485	3032	117	137	342	14.1
APRES	91	72	13	23		12.7
ASCAL	3397	1579	76	87	176	
MXMAP	191	162	19	30	118	14.6
MXCLN	9920	5204	170	204	709	14.0
VM	280	259	46	68	223	23.9

Table 2
Selected ratioes

Task	15JUL85/ 15JUL86 PSAP(cpu)	VAX/ Convex PSAP(cpu)	VAX AP/ Convex (real)
UVMAP	1.3	6.0	2.8
APCLN	1.5	25.9	2.5
APRES	1.3	5.5	
ASCAL	2.2	20.8	2.5
MXMAP	1.2	8.5	3.9
MXCLN	1.9	30.6	3.5
VM	1.1	5.6	3.3

Notes:

1) A new residual packing scheme for the AP caused the APCLN and MX to take different paths in the 15JUL85 and 15JUL86 versions; the 15JUL86 MXCLN took an extra major cycle.

2) The major improvement in ASCAL and MXCLN in the 15JUL86 VAX Pseudo AP version is the table lookup polar to rectangular conversion routine.

3) The first column in table 2 is the cpu time ratioes for the old to new pseudo AP on the VAX. The second column is the cpu ratioes of the new (15JUL86) pseudo AP on the VAX to the Convex. The third column is the ratio of real times on the VAX+AP to the Convex.