AIPS MEMO NO. 38

Certification and Benchmarking of AIPS on the Convex C-1 and Alliant FX/8

Kerry C. Hilldrup, Donald C. Wells, William D. Cotton

National Radio Astronomy Observatory [1]
Edgemont Road, Charlottesville, VA 22903-2475
(804)296-0211, FTS=938-1271, TWX=910-997-0174

24 December 1985 [2]

Abstract

The 15APR85 release of AIPS has been installed on the Convex C-1 vector computer and on the Alliant FX/8 vector/concurrent computer. Both have been certified using the PFT benchmarking and certification test. Although a small number of compiler bugs were encountered in each, the AIPS application code was installed with only minimal modifications, and computed results agreed well with other implementations of AIPS. In the course of the implementations, the technology of the Clark CLEAN algorithm was advanced considerably; the final vectorized CLEAN algorithm on both systems is about three times as fast as the current microcode algorithm on the FPS AP-120B array processor. Similar improvements were observed for other highly vectorized tasks. Programs which were not vectorized generally executed on both in comparable CPU times and faster real times than they achieved on the DEC VAX-8600. The FX/8 with 6 computational elements (CEs) generally outperformed the C-1 by a little in CPU time, but was significantly slower in real time. The performance of the FX/8 as a function of the number of CEs is also described.

---

[1] NRAO is operated by Associated Universities, Inc., under contract with the U. S. National Science Foundation.

[2] this report entirely supersedes and replaces a previous version which was dated 18 July 1985 and was marked as "AIPS Memo No. 37".

CONTENTS

1 INTRODUCTION

NRAO tested both the Convex C-1 vector computer and the Alliant FX/8 vector/concurrent computer as candidates for the replacement of the IBM and Modcomp systems in Charlottesville. The goals of the tests were to assure that these candidate systems would execute AIPS properly, and to assess their performance relative to each other and to other machines, including the VAX-11/780s which NRAO already owns and the VAX-8600, another candidate for the procurement [3].

2 ABOUT THE CERTIFICATION AND BENCHMARKING PACKAGE

> "[our evaluation] was not designed to be a 100-yard dash for computer systems... it was intended as a decathlon..." [4]

NRAO's programming group in Charlottesville has a standard procedure for assessing a computer system for use with AIPS: install AIPS ("Astronomical Image Processing System") on it and run the AIPS certification and benchmarking test package. Successful execution of the test certifies that the computer hardware, the FORTRAN compiler, the operating system components, and the interface between AIPS and the operating system (the "Z-routines") all behave correctly. Benchmarking data can be extracted from the time stamps recorded in the AIPS message file and from the accounting listings produced by the AIPS utility PRTACC. Since the test procedures are written in the AIPS command language called POPS ("People-Oriented Parsing System") and read a binary data tape written in FITS ("Flexible Image Transport System") format, they are inherently machine- and operating system- independent. In a procurement situation, the aim is to perform exactly the same test on all of the machines which are under consideration. Note that this does not mean that all of the code of the test is the same on all machines. Only the portable portions of the application programs, the POPS procedures, and the FITS files must be invariant; it is not only permissible but even desirable that the system-dependent portions of AIPS should be customized to achieve the best performance on each separate host system.

The tests discussed in this memo were all performed with an experimental version of the test package, which was named "PFT" (the production release of the package will be similar, but will be called "DDT"). The package consists of two command language scripts, called "RUN files" in AIPS parlance (a total of about 400 lines of text), and a tape containing "master" data and comparison images. The first script compiles the test procedures and the second script executes them. The PFT test executes twelve different AIPS programs (AIPS,

---

[3] see AIPS Memo No. 36, "Certification and Benchmarking of AIPS on the VAX-8600", 24 June 1985.

[4] R.P.Colwell, C.Y.Hitchcock, E.D.Jensen, and H.M.Brinkley Sprunt, in "Open Channel", Computer, Vol. 18, No. 12, December 1985, p. 93.

IMLOD, UVLOD, UVSRT, UVMAP, COMB, APCLN, SUBIM, ASCAL, MX, CNVRT, and VM). These programs, often referred to as the "Dirty Dozen", are used to process a real dataset of fringe visibilities collected with NRAO's Very Large Array (VLA) telescope. At each step where an image is computed, it is compared against the respective master image and residuals are summarized. A full description of the details of the test package is beyond the scope of this report; it is sufficient to say that the package reasonably reflects the actual use of AIPS on real data. Some parts of it are I/O limited, some are CPU-bound, and some are sensitive to various sources of overhead such as creating, opening, truncating, closing, and cataloging files. The CPU-bound tasks test a diverse range of heavy vector computing problems.

The statistics obtained from comparing the master and computed PFT answers must be construed with care. Some residuals are due to differences in floating point hardware, others are due to differences in the algorithms in various releases of AIPS, and some residuals may actually indicate errors of implementation on the machine being tested. In particular, maps computed in a host computer will almost always give non-zero residuals when they are compared to those computed with an FPS AP-120B, because the AP uses 28 bit floating point fractions, while 32-bit hosts (e.g., VAX, C-1, FX/8) compute with 24 bit fractions. This situation occurred in the present trials, because the master data files for the test were generated on a VAX-780 using an AP-120B. The AIPS certification procedure is not concerned with assuring that an algorithm tells the "Truth" about the sky; rather, its goal is to ascertain whether two computers "tell the same lie", within an acceptable tolerance.

3 A HISTORY OF NRAO'S 1985 BENCHMARKING CAMPAIGN

Two of the authors (KCH & DCW) spent two days, 30-31 May, at Convex Computer Corporation's factory in Richardson, TX, to make the initial installation of AIPS on the C-1. One particular compiler bug involving computed-GO-TOs with INTEGER*2 variables caused a delay of about 24 hours in the installation process, but, by the end of the second day, five of the twelve programs tested by PFT had passed the test. The initial installation did not exploit the vector computing capability of the C-1. During the next three weeks the authors worked through dialup modems to track down the remaining compiler bugs and AIPS bugs, but mostly to significantly improve the vectorization of the pseudo array processor library. By the 20th of June the C-1 had demonstrated speed 2-3x faster than the FPS AP-120B on several AIPS tasks, and the entire dozen tasks of the PFT test had passed the certification test.

The Unix version of AIPS which was installed on the C-1 was very similar to the 15APR85 release of AIPS, which was the version used to test the VAX-8600 on 29 April. This Unix AIPS version was copied from 15APR85 on 5 February, two weeks before 19 February when the master test case of PFT was computed. The reason for using the February version of the test package for these tests rather than more recent

versions was not only that it matched the AIPS version, but also that the February test tape has been used to test a number of different machines during 1985 and it is desirable to be able to intercompare all results.

Early in July, NRAO learned of the existence of Alliant Computer Corporation and their FX/8 computer. One of the authors [DCW] attended the announcement for this machine on 24 July in Boston, and spent 25 July at the factory in Acton, MA. Early in August, the entire set of AIPS files was transported from the Convex C-1 to the Alliant FX/8, and was then modified to run on that machine. There was approximately a ten day delay in the installation due to an inability of Fortran modules to call Z-routines written in C-language. Fortunately, a library of Fortran-callable routines was available that, in many cases, could be used to serve the same purpose as the otherwise unreachable C library functions. For the remainder, Alliant provided an assembler interface (or "wrapper") routine which could be cloned to call the C-language routines distributed as part of the standard AIPS installation kits for Unix systems. Several minor bugs, again involving INTEGER*2 variables, were uncovered in the compiler and fixed. Essentially all of AIPS was then compiled with the optimizer fully enabled. By late August AIPS was operational on the FX/8, and had passed the PFT test.

It was obvious that both the C-1 and the FX/8 were viable candidates for the procurement, and that a controlled comparison was required. Accordingly, late in August NRAO resumed the Convex tests; after the last few compiler bugs were found, the September version of the Convex compiler was able to compile essentially all of AIPS with the highest level of optimization enabled. The major change in the Convex operating system between June and September was the availability of "disk striping" [5]. Also, the C-1 tested in September had a 9.5 MHz clock (the June machine was 9.0).

During the first two weeks of September, the Alliant FX/8 and Convex C-1 systems were compared in detail, simultaneously, using the same application code and the same certification and benchmarking test kit as had been used to test the VAX-8600 in April. Both vector machines passed the certification test, and both were accepted as viable candidates for the procurement. The final trials for the purpose of the procurement were run on 13 September; the results are shown below in Tables 1 and 2, and discussed in Section 4.3. The basic

---

[5] The term "striping" refers to DMA (direct memory access) transmissions from two or more disk controllers reading into a common (double) buffer concurrently. The sectors of the disk file are divided among the available drives; for a two-disk stripe, sectors 1,3,5... are on the first drive and 2,4,6... are on the second. Simultaneous interleaved reading or writing of the sectors with independent DMA controllers will double the transfer rate, thus synthesizing a double-performance disk out of two lower performance devices. The Convex C-1 tested in September was configured with a four-disk stripe, which quadrupled performance.

procurement decision was made soon after the completion of those trials.

Subsequently Alliant made certain changes in their I/O system which they believed would improve performance, and they asked NRAO to rerun the test suite as a courtesy (not for the procurement evaluation). The final runs of this set of tests were made on 24 November; they are summarized below in an appendix.

4 THE AIPS PFT BENCHMARKING RESULTS

The trials of the Convex and Alliant machines were made with essentially the procurement configurations. In the case of the Convex, this meant 32 MB of memory and a four-way striped disk system. For Alliant, it meant 32 MB of memory and up to 8 CEs [6], with sufficient IP and disk capacity (the NRAO proposed procurement configuration was 6 CEs). The results of the "normal" PFT trials appear in Table 1 below, along with comparison measurements made on a number of other machines (780, 8600, 780+AP, FX/1 [7], and Cray X-MP). The first three data columns (780, 8600, and 780+AP) of Table 1 below are taken from the corresponding table in AIPS Memo No. 36 (see footnote [3]).

4.1 Notes For The "Normal" PFT Trials

When examining the tables below the reader will want to keep the following facts in mind:

1. Tests showed that when the output being sent to the terminal was sent to the Unix "null" device instead, the real times decreased significantly, demonstrating that performance was degraded by terminal I/O limitations (all tests were made with telephone lines at 1200 baud). The real-time numbers in Table 1 do not include any corrections for this phenomenon; they are the actual values measured in the formal PFT trials.

---

[6] Alliant's "FX/8" model is composed of 1-8 separate, identical "computational elements" (CEs). In this report the notations FX/1, FX/2, FX/4, FX/6, FX/8, and FX/n refer to the FX/8 model with 1-8 CEs.

[7] The FX/1 tested by NRAO is an FX/8 configuration with only 1 CE active. Alliant's "FX/1" model is a special configuration which can only have one CE, and which has a different arrangement of cache memory from the FX/8. The authors assume that the CPU performance of the two 1-CE configurations is approximately equal, and that I/O performance is mainly determined by the peripherals, rather than the CPU. Therefore, NRAO's "FX/1" measurements should be useful as an approximate guide to the performance of the real FX/1.

2. It should be noted that the 780 and 8600 AIPS installations make use of "asynchronous I/O" [8], whereas this feature was not available in either the C-1 or the FX/n at the time of the trials.

3. The CPU/Real ratios in Table 1 are all just the CPU time in that table divided by the corresponding real time (note that when an AP is involved, the CPU time includes no contribution from the AP itself except for the handler). The ratios indicate the extent to which a task exhibits un-overlapped I/O or operating system overhead. Some tasks, a good example is MX with pseudo-AP on the 780, manage to almost completely hide heavy I/O activity behind their CPU operations, whereas others, VM and UVMAP for example, have rather low CPU/Real ratios. Some tasks are I/O dominated and exhibit quite low ratios; UVSRT and SUBIM are good examples (UVSRT does a disk merge sort and SUBIM is effectively a file copy operation). I/O dominated tasks like CNVRT, COMB, SUBIM, and UVSRT will gain little if any advantage from vectorization or concurrency.

4. Tasks APCLN, APRES, ASCAL, MXMAP, MXCLN, UVMAP, and VM are the "AP-tasks" in the PFT trial. They all gain directly from vectorization or concurrency.

5. Task ASCAL is the most highly vectorized AIPS task; it is also especially indicative of vector sine/cosine performance. Note that the FPS 120B array processor uses special lookup tables to evaluate sines and cosines; this makes it unusually effective for ASCAL, compared to its peak floating point pipeline capability.

6. Task VM (maximum entropy image deconvolution) only uses the "AP" for 2-D FFTs; the timings in Table 1 under-represent its ultimate performance on these machines.

7. The authors consider the MXCLN timings to be the best single AIPS performance index, because task MX in its cleaning (deconvolution) mode does a little bit of everything (gridding, transforming, cleaning, and UV-subtracting).

---

[8] The term "asynchronous I/O" refers to DMA (direct memory access) transmission from a disk controller into a double buffer in the address space of the program, with the transmission occurring concurrently with CPU execution in other parts of the address space; this gives optimum real-time and CPU-time performance. Unix systems traditionally offer only "synchronous I/O", in which execution of a program stops while any I/O transmission into the program's address space is in progress; this degrades real-time performance. In addition, Unix systems traditionally use a form of "move-mode I/O" in which all transmissions are made by copying from a buffer in the operating system's space to the buffer in the program's space; this degrades CPU-time performance, even though read-ahead and write-behind in the OS buffer helps real-time performance.

4.2 The "Scaled-Up" PFT Trials

The basic PFT test is a modest synthesis mapping problem in the context of the VLA (of course, any problem which consumes 19,000 seconds on a VAX-780 is hardly a "modest" problem in an absolute sense). During the course of the September trials the question arose of whether important performance differences between systems would appear in a much larger-sized problem. Practical limitations prevented the processing of larger datasets on remote machines, but the AIPS tasks could be ordered to do more work on the available dataset. This was done, by making minor modifications in the command language test procedures. The resulting measurements appear in Table 2 below.

4.3 Conclusions Drawn From The PFT Measurements

The following conclusions can be drawn from these data:

1. The VAX-8600 is 4-5x faster than the 780 in CPU-time (but I/O-limited tasks are only slightly faster in real-time because the two machines use the same disks), and the C-1 and the FX/1 are about equal to the 8600 in scalar performance.

2. For AP-tasks, the 780+AP is 8-14x faster than the 780, and the C-1 and FX/6 are about 3x faster than the 780+AP.

3. The C-1 disks are about 3x faster than the 8600 disk; the authors speculate that this performance advantage is due to the use of disk striping in the C-1. Note the high CPU/Real ratios exhibited by the C-1 (total across the PFT test suite of 90%).

4. The X-MP uniprocessor is approximately 5-10x faster than the C-1 and FX/6 for AIPS computing. The numbers shown for the X-MP represent the performance of NRAO's X-MP/COS implementation as it stood in mid-September 1985. The PFT run was made during mid-day on a loaded machine; therefore any intercomparison of real times is meaningless. Both the overall performance of the X-MP/COS implementation, and the performance of individual tasks, have been somewhat enhanced since that time.

5. Regarding the C-1 versus FX/6 comparison, there is no easy, clearcut technical choice:

   - The FX/6 is generally faster than the C-1 in CPU-time,

     but

   - the C-1 is generally faster than the FX/6 in real-time.

   - In the "scaled-up" problem, the CPU-time advantage of the FX/6 was mostly attenuated, while the real-time advantage of the C-1 became even more pronounced. Note that the FX/6 remained superior for ASCAL; indeed its lead increased for this task. This illustrates the high performance of concurrent-scalar transcendental operators in the FX/8 (see Section 7.4 below).

NOTE

It is obvious that various ratios between the computing performance of the C-1 or FX/6 and that of the 780 or 8600 can be derived from the data in the tables. For example, the MXCLN problem used 9921 seconds CPU time on the 780, and only 221 on the C-1, for a speed ratio of 44.9x. Or, ASCAL ran 1058 seconds on the 8600, but only 81 on the FX/6, for a ratio of 13.1x. It is tempting to assume that such ratios will apply to all engineering-scientific computing applications, but the authors caution that these ratios should not be quoted out of their context. The reason is twofold: (1) fundamentally they only apply to the specific algorithms of AIPS, and only to the specific implementations of AIPS that were made during the limited period of time during the summer of 1985, and (2) for the pure scalar machines, they represent the speed achieved by the publicly distributed versions of the Q-routine library for the 15APR85 release of AIPS, not the speed which the vectorized CLEAN would have if it executed on the scalar machines.

Table 1

Alliant and Convex PFT Benchmarks, "Normal" Problem,
with comparison to several other machines,
as of mid-September 1985

		780		8600		780 + AP		FX/1		FX/6		C1 <6>		FX6/C1		CRAY X-MP	
TASK	IOCNT	REAL	CPU	REAL	CPU	REAL	CPU	REAL	CPU	REAL	CPU	REAL	CPU	REAL	CPU	REAL<3>	CPU
AIPS	380	?	?	5371	?	2306	?	<2>	131.2	1158	128.1	845	117.5	1.37	1.09	5503	23.1
CNVRT	25	8	5.9	4	1.7	8	5.9	4	2.1	4	1.4	3	1.9	1.33	.74	4	.2
COMB	33	32	22.7	13	4.9	32	22.7	9	6.9	8	5.7	6	5.4	1.33	1.06	11	1.0
SUBIM	19	14	6.4	13	2.1	14	6.4	6	2.8	4	2.3	3	2.1	1.33	1.10	6	.4
UVSRT	96	39	20.8	27	5.1	39	20.8	25	10.1	25	9.0	9	6.6	2.78	1.36	35	1.0
APCLN	267	4577	4486.0	963	931.0	340	144.4<1>	307	264.6	134	86.6	133	122.3	1.01	0.71	160	21.2
APRES	82	113	91.8	35	23.7	60	32.5<1>	28	17.6	24	12.2	13	10.1	1.85	1.21	102	1.4
ASCAL	136	3509	3398.0	1088	1058.0	154	35.2<1>	410	394.8	91	80.5	107	103.7	.85	.78	54	11.1
MXMAP	96	222	191.9	71	47.7	106	47.4<1>	69	51.8	38	21.9	31	27.2	1.25	.81	112	2.1
MXCLN	481	10033	9921.0	2340	2275.0	712	228.3<1>	639	573.3	262	186.8	236	221.0	1.11	.85	833	24.9
UVMAP	151	170	151.9	59	39.7	81	41.5<1>	62	37.6	45	17.8	23	18.3	1.96	.97	104	2.0
VM	344	320	280.9	116	71.6	195	126.7<1>	125	64.4	88	41.5	45	30.0	1.96	1.38	653	3.9
ALL<4>	1730	19037	18577.3	4729	4460.5	1741	676.6<1>	1684	1426.0	723	465.7	609	548.6	1.19	.85	2074	69.2
		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL				CPU/REAL	
AIPS		N/A		N/A		N/A		N/A		N/A		N/A				N/A	
CNVRT		.74		.43		.74		.53		.35		.63				N/A	
COMB		.71		.38		.71		.77		.71		.90				N/A	
SUBIM		.46		.16		.46		.47		.58		.70				N/A	
UVSRT		.53		.19		.53		.40		.36		.73				N/A	
APCLN		.98		.97		N/A		.86		.65		.92				N/A	
APRES		.81		.68		N/A		.63		.51		.78				N/A	
ASCAL		.97		.97		N/A		.96		.88		.97				N/A	
MXMAP		.86		.67		N/A		.75		.58		.88				N/A	
MXCLN		.99		.97		N/A		.90		.71		.94				N/A	
UVMAP		.89		.67		N/A		.61		.40		.80				N/A	
VM		.88		.62		N/A		.52		.47		.67				N/A	
ALL<4>		.98		.94		.60<5>		.85		.64		.90				N/A	

<1> represents VAX/780 CPU time only; no accounting information for AP CPU time; use REAL times for comparison
<2> tasks initiated standalone to insure a 1 CE environment; AIPS real time is not meaningful. Compiled "vector-only".
<3> loaded machine; typical for mid-afternoon
<4> tasks only (i.e., does not include AIPS itself)
<5> for non-AP tasks only
<6> Convex C-1 executing at 9.5 MHz, with 4-way disk stripe.

Table 2

Alliant and Convex PFT Benchmarks, "Scaled-Up" Problem

same number of visibilities; cellsize decreased from 8" to 2" (mapsize up from 256 to 1024); CLEAN loop gain decreased from 0.1 to 0.01; number of components increased from 2000 to 4000; all task loadings are increased, with the exception of UVSRT. Measurements made mid-September 1985.

		FX/6			C1			FX6/C1	
TASK	IOCNT	REAL	CPU	RATIO	REAL	CPU	RATIO	REAL	CPU
AIPS	712	12513	386.1	N/A	6554	613.8	N/A	1.91	.63
CNVRT	93	42	9.4	.22	24	22.2	.93	1.75	.42
COMB	1466	110	61.2	.56	66	60.7	.92	1.67	1.01
SUBIM	105	51	9.9	.19	15	13.0	.87	3.40	.76
UVSRT	96	29	9.2	.32	8	6.7	.84	3.63	1.37
APCLN	16743	4320	1364.2	.32	1795	1199.3	.67	2.41	1.14
APRES	1276	317	124.6	.39	146	108.2	.74	2.17	1.15
ASCAL	136	160	149.1	.93	221	217.4	.98	.72	.69
MXMAP	1537	324	149.2	.46	214	163.6	.76	1.51	.91
MXCLN	13351	3520	1832.8	.52	2057	1536.9	.75	1.71	1.19
UVMAP	1466	435	138.7	.32	174	141.2	.81	2.50	.98
VM	7781	1843	569.6	.31	671	406.9	.61	2.75	1.40
ALL<1>		11151	4417.9	.40	5391	3875.9	.72	2.07	1.14

<1> tasks only (i.e., does not include AIPS itself)

5 VECTORIZATION ISSUES

AIPS was originally developed to be used with Floating Point Systems array processors, models 100, 120B, 5105, and 5205. For a host machine that does not have an AP, the application tasks are linked against a library of FORTRAN subroutines which have the same names, arguments, and functionality as the library of microcode routines for the FPS AP. These "pseudo array processor" routines (also called the "Q-routines") emulate operations which execute in a vector manner in the FPS APs; it follows that they are candidates for vectorization in vector machines like the Convex C-1, Alliant FX/8, and Cray X-MP. Because many of the major application tasks of AIPS have been deliberately designed to do as much of their computing in the AP as is possible, it follows that vectorization of the Q-routines effectively vectorizes most of the heavy computing of AIPS.

For a vector processor, AIPS performance can be increased by modifying several of the routines in the Q-routine library. In particular, the original FORTRAN formulation of the Clark CLEAN algorithm in the library proved to be slow on the C-1 in June. Two of the authors (DCW & WDC) developed a new formulation of the algorithm for vector computers during the course of the C-1 installation. The algorithm was also used in the Alliant FX/8 and Cray X-MP installations, with appropriate machine-dependent customizations. This new CLEAN algorithm is believed to be applicable to other vector machines in the commercial market (Cray-2, CDC/ETA, Fujitsu/Amdahl, IBM...).

Vendor-supplied FFT subroutines were used in the Convex, Alliant, and Cray installations. The new CLEAN algorithm depends on the ISAMAX subroutine from the LINPACK linear algebra library, and a vendor-supplied version of ISAMAX was available on each machine. The CLEAN also uses the "WHENILT" library routine of the Cray library. Assembly language versions of WHENILT were prepared by one of the authors [DCW] for both the Convex and Alliant implementations; each was subsequently "tuned" by vendor personnel.

Although most of the heavy computing of AIPS is concentrated in the vectorized Q-routines, "Amdahl's Law" does apply: the performance of parallel machines is ultimately limited mainly by the fraction of the code which executes in non-parallel mode [9]. Many DO-loops in AIPS programs and subroutines did not vectorize initially due to references to equivalenced variables, or due to use of subscript offsets whose values are not known at compile time. Vectorizing compilers refuse to vectorize such cases of apparent dependency in the code. The most common cure is to introduce compiler directives into the text of the subroutines, especially the "NO_RECURRENCE" directive (this is the Convex name; it is called "IVDEP" by Cray and "NODEPCHK"

---

[9] see p. 11 of: O.Lubeck, J.Moore, and R.Mendez, "A Benchmark Comparison of Three Supercomputers: Fujitsu VP-200, Hitachi S810/20, and Cray X-MP/2", Computer, Vol. 18, No. 12, December 1985, pp. 10-24.

by Alliant). The appropriate directives are now INCLUDEd before all DO-loops in the Q-routine library which vectorizing compilers suspect may contain vector dependencies, but which in fact do not. In some cases the code in DO-loops only needed to be rearranged to permit greater vectorization. In general, almost any DO-loop anywhere in AIPS stands to benefit from such changes. Because the C-1 and FX/8 architectures support vector operations on 8, 16, and 32-bit integers as well as floating point, the authors expect that even the processing of FITS tapes is capable of at least some vectorization.

6 THE PROCUREMENT DECISION

A good summary of the outcome of NRAO's 1985 procurement trials is:

> "Three systems were qualified for the procurement using the AIPS PFT certification and benchmarking package: DEC VAX-8600, Convex C-1, and Alliant FX/n. The test results show that the latter two systems are clearly superior in performance to the 8600. Both the Alliant and the Convex systems make fine AIPS machines, and the AIPS Group recommends both of them to any group which wants a high-performance AIPS computer based on state-of-the-art technology. The two systems were judged to be roughly equal on overall technical grounds at this particular date and for this particular procurement, so our final procurement decision will likely depend on issues other than performance alone.... Regardless of which of the two vendors is selected for NRAO's procurement, the AIPS Group stands ready to assist any AIPS site which subsequently chooses either vendor. We also recognize that NRAO's evaluation may become obsolete as the relative price, performance, and features of these relatively new machines evolve in the coming months." [10]

The "particular date" for this procurement was approximately 16 September 1985. Both the Convex C-1 and the Alliant FX/n have special features and exhibit superior performance in particular cases, a situation which can make either of them preferable for specific purposes. Unfortunately, it was necessary to select only one of them for this procurement; NRAO chose the Convex C-1, and installation was scheduled for the last week of December 1985.

---

[10] AIPSLetter, Vol. V, No. 4, 15 October 1985, pp. 1-2.

7 APPENDICES

7.1 Reviews Of The Two Machines

The operating systems of both the Convex and the Alliant are based on the Berkeley virtual memory version of Unix, commonly known as "4.2bsd". The authors made no quantitative measures of the effect of timeshared loading on performance, but formed the impression that the character of the effects is roughly linear on both machines. Both systems support the DOD/ARPA TCP/IP network protocol on Ethernet LANs.

Both compilers are robust, sophisticated products. Both implement most of the VMS extensions to Fortran-77, and both attempt to make the same interpretation of the language that the VMS compiler does, as far as possible. Both are capable of vectorizing DO-loops which contain IF-statements. The sophisticated vectorizing compilers make both systems attractive for general engineering and scientific computing, not just for AIPS. In general, both are much easier to use than array processors.

7.1.1 Convex C-1

The C-1 is a "vector register" machine; its architecture, which is implemented in custom CMOS gate-arrays, resembles that of the Cray-1 in many respects [11]. The 8 vector registers are 128 words long; supported data types are 8, 16, 32, and 64-bit integers, and 32 and 64-bit floating point (using the VAX "F" and "G" formats). If only the floating add and multiply operators are counted, the maximum speed of the C-1 is 40 MFlops for 32-bit floating data and 20 MFlops for 64-bit. Not only the scalar units but also the vector pipelines operate on all of the data types, with 8 and 16-bit data being processed at 32-bit rates. The vector pipelines can be "chained", as in the Cray, and scalar operations can overlap vector pipeline activity. The scalar speed of the C-1 is approximately equal to that of the DEC VAX-8600.

The vector hardware architecture of the C-1 includes the full complement of advanced vector operators: gather, scatter, comparison, mask, compress, and merge. In addition, the C-1 includes a nice set of vector reduction operators: SUM, PRODUCT, MAX, MIN, AND (ALL), OR (ANY), and XOR (PARITY); reduction operations reduce a vector to a scalar. For example, the sum-of-vector enables the C-1 to chain the dot product of two vectors: a vector multiply instruction immediately followed by the sum-of-vector accumulates the dot product in one of the

---

[11] see T. Jones, H.W. Dozier and J. Gruger, "Supercomputer Breaks Price Barrier for Vector Processing", Computer Design, April 1985, pp. 169-176.

scalar registers at the maximum rates quoted above. Two more essential capabilities are provided: population count and trailing zero count. The first counts how many comparisons succeeded (i.e., how many bits are set in the vector mask register) and the second is used to find the first occurrence of a successful comparison in a vector. The compress, merge, pop count, and trailing zero count operations are not directly expressable in FORTRAN, but they are accessible through the assembly language. A programmer can hand-code critical portions of algorithms in assembly language subroutines in order to use these advanced vector operators; in this way the ultimate performance of the C-1 is available.

The C-1 is a full virtual memory machine; addressing is "non-byte-swapped". Current chip technology (256K chips) permits 128 MB physical memory size. The CPU-to-Memory bandwidth is 80 MB/sec. Memory is dual ported, and the I/O system has concurrent bandwidth to memory of 80 MB/sec, for a total of 160. The I/O interface is IEEE-standard Multibus. In order to get sufficient aggregate I/O performance, multiple Multibusses are operated concurrently. The Multibusses are connected to the I/O system through up to five "IOPs" (I/O Processors), each of which is computer in its own right; the device drivers execute in the IOPs, thus reducing interrupt overhead on the main CPU. Each IOP can control up to four Multibusses.

The Convex C-1 was announced in the fall of 1984. List prices begin at about US$500K, and prices for typical large configurations are in the neighborhood of $750-800K. For further information, contact:

Convex Computer Corporation (214) 952-0200
701 Plano Road
Richardson, Texas 75081

7.1.2 Alliant FX/8

The FX/8 contains from 1 to 8 "computational elements" (CEs), each of which is both a scalar and vector computer approximately equal in power to a VAX-8600 plus AP-120B array processor [12]. The scalar instruction set is a CMOS gate-array implementation of the Motorola 68020 architecture, with vector register and concurrency instructions added. The scalar floating point instructions include several transcendental operations (square root, sine, cosine, logarithm, and exponential). Bit and bit-field instructions are available.

The eight vector registers of an Alliant CE are each 32 words long. Supported data types are 8, 16, and 32-bit integers, and 32 and 64-bit floating point (using IEEE-standard formats). An FX/8 system

---

[12] see S. Lackey, J. Veres and M. Ziegler, "Supercomputer Expands Parallel Processing Options", Computer Design, 15 August 1985, pp. 76-81.

with 8 CEs has a peak performance rating for 32-bit data of about 35 MIPS for scalar-concurrent operations and 94 MFlops for vector-concurrent computing (4.45 MIPS and 11.8 MFlops per CE). For 64-bit data, the numbers are about 29 MIPS and about 47 MFlops. In general, scalar operations do not overlap vector pipeline activity.

The vector instruction set is richer than that of the C-1 (or the Cray-1), because operator combinations are implemented explicitly, rather than depending on the "chaining" technique to synthesize them. Like the C-1, the FX/8 architecture includes the full complement of advanced vector operators (gather, scatter, comparison, mask, compress, and merge) and a nice set of vector reduction operators: DOT, SUM, PRODUCT, MAX, MIN, AND, OR, and EOR. The FX/8 also supports a "polynomial" reduction operator. Population count, leading zero count and trailing zero count are available for the vector mask register. The mask register is effective in almost all vector instructions, like the Fujitsu VP-series and unlike the Crays and the C-1. As with the C-1, the full vector architecture can be made available to Fortran programs by coding assembly language subroutines.

Up to twelve 68012 computers are used as "interactive processors" (IPs) and peripheral controllers (each IP controls one Multibus). The IPs have a separate path to main memory, and they generally execute the device drivers and timesharing utility operations; one of the IPs executes the operating system code to manage queues for the 8 CEs and 11 other IPs. Note that pure-scalar code can execute on either a CE or an IP.

Alliant's FX/1 model contains one CE (4.4 MIPS, 11.8 MFlops), and either one or two IPs. Cache memory is shared, and because there is only one CE, the concurrency logic is not used (the compiler can be instructed to compile vector-only code to reduce overheads).

The FX/8 and FX/1 are full virtual memory machines (2 GB virtual space, in both CEs and IPs); like all 68000s, addressing is "non-byte-swapped". Current chip technology (256K chips) permits up to 80 MB physical memory size in the FX/8. The main memory has a total bus bandwidth of 188 MB/sec, which support two paths at 94 MB/sec each, one for the IPs and their I/O devices and one for the CE complex.

Alliant's Fortran compiler automatically generates code to exploit the vector and concurrent architecture; concurrent code automatically utilizes as many CEs as are available. This compiler represents a major technical achievement: it is the first commercially available Fortran compiler which supports parallel execution automatically. Special compiler directives are provided for the programmer to control compilation modes for loops. The compiler also implements the vector notation extensions which have been proposed for Fortran-8x.

The concurrent execution of the CE complex depends critically on a special hardware "concurrency bus" which interconnects the CEs. It permits starting, stopping, and synchronizing CEs on a microsecond timescale. Even dependent execution of loop iterations is supported by this hardware. Variables in shared memory can also be used for

intercommunication between CEs (using the coherent cache) and for synchronization (the 68000 "test-and-set" instructions work well for this purpose).

The Alliant FX/8 and FX/1 were announced in July 1985. List prices for the FX/1 start below US$150K; prices for the FX/n range from about $270K to about $950K. For further information, contact:

Alliant Computer Systems Corporation (617) 263-9110
42 Nagog Park
Acton, Massachusetts 01720

7.2 Suggestions For Improvements

One duty of any benchmarking project is to point out areas where improvements are possible. Suggestions which apply to both vendors will be covered first, followed by items specific to each vendor.

7.2.1 Suggested Improvements For Unix -

1. Ability to pre-allocate space for a disk file at the moment of creation is desirable, and is not a standard feature of Unix. It is unpleasant to compute for an hour generating an output file, and then crash due to insufficient space to complete the file!

2. To complement pre-allocation, a means of truncating files is highly desirable, and is not a standard feature of Unix (Convex provides this capability via the function "ftruncate", an extension to the 4.2 bsd UNIX C library). Otherwise, file truncation can only be performed under Unix by copying, deleting and renaming.

3. The ability of a process to obtain exclusive access to a file is desirable, and is not a standard feature of Unix (it is "flock" under 4.2bsd and "lockf" under System V; when will they agree?). "Lock files" are a mediocre substitute for this fundamental functionality.

4. Support for asynchronous device I/O is desirable, and is not a standard feature of Unix. Both disk and tape asynchronous I/O should be supported. In addition to read and write operations, asynchronous tape operation should return block lengths and should also support tapemark, BOT and EOT sensing, as well as record and tapemark skipping (unload capability would be nice).

5. Support for "mapped (virtual) file I/O" is ultimately desirable for some applications, although not necessarily for AIPS. It is implemented in several modern operating systems, but was not included in the Berkeley virtual memory Unix design.

6. A notable weakness of extant Unix implementations is that the available debuggers appear to be inferior to the VAX/VMS debugger (especially the VMS 4.0 version). Unix vendors would do well to study, and perhaps emulate DEC's implementation.

7.2.2 Fortran Compiler Improvements -

1. Inability of the Convex and Alliant compilers to optimally vectorize a key DO-loop containing an IF forced the authors to resort to special coding techniques (WHENILT routine coded in assembly language) to get proper performance in the CLEAN algorithm. There are three possible ways to vectorize IF-statements: masked arithmetic, compress/expand, and gather/scatter (which the authors used to vectorize the CLEAN algorithm). The Convex and Alliant compilers currently only support one of these, masked arithmetic. By comparison, the Fujitsu compiler supports all three, and apparently can automatically produce nearly optimum code [13]. The Convex and Alliant compiler groups should consider implementing at least the gather/scatter scheme for IF-statements, with activation controlled by a compiler directive. The most elegant approach would be to implement a "truth ratio" compiler directive, as Fujitsu does, and let the compiler decide (automatically) how to compile the loop optimally.

2. Compilers should vectorize expressions containing equivalenced variables if the programmer asserts the no-dependency directive.

3. It would be a good thing if all vendors of vectorizing compilers would agree on the format and meaning of the common compiler directives. For example, is there any good reason for Cray's compiler to use "IVDEP" while the Convex compiler wants "NO_RECURRENCE"? Standardization of directives would be a good goal for the ANSI X3J3 (Fortran) committee; perhaps Convex and Alliant could join with other vendors to implement this.

4. If a DO-loop has a variable for its limit, a vectorizing compiler is forced to assume that the run-time limit may be larger than the vector register size. If the loop limit is, in fact, smaller than the register size, the long-vector logic is useless, and execution of these instructions may degrade performance. Convex and Alliant should consider implementing a "SHORTLOOP" directive analogous to the one offered by the Cray CFT compiler in order to eliminate the useless "strip mining" logic. This may be particularly important for Alliant, because less of the scalar overhead is hidden behind

---

[13] see "Facom Vector Processor System: VP-100/VP-200", by K. Miura and K. Uchida, pp. 59-73 of "Supercomputers: Design and Applications", ed. K. Hwang, IEEE Computer Society, 1984, LOC=QA76.5.T88, ISBN=0-8186-0581-2.

the pipes in the FX/n.

7.2.3 Specific Suggestions For Convex -

1. Can the vector sine and cosine routines be improved? NRAO's measurements suggest that the present sine/cosine routines are little if any faster than those of the FPS AP-120B. Perhaps a table-lookup approach, analogous to that in the 120B, could make a useful gain?

2. NRAO's measurements showed that about 6% of the cost of executing MX went into "vector" fix and float operations (subroutines _mth_$vj_cvt_vr and _mth_$vr_cvt_vj); apparently these routines actually execute in scalar mode. Perhaps it would be possible to add one or more opcodes to the vector pipelines in order to produce a higher performance implementation of these functions?

7.2.4 Specific Suggestions For Alliant -

1. Implement a Fortran-callable interface for C procedures.

2. Implement disk striping.

3. The Alliant architecture is complicated (both vector registers and fine-grained concurrency); there is usually more than one way to code any algorithm, and the complexity means that it is not always obvious which way is best. In fact, it appears that the best scheme generally depends on the vector lengths and the number of CEs currently available in the complex. One way to optimize coding would be to generate code for each scheme, and make a real-time decision. Memories are now large; the extra code is less important than the extra performance enhancement. The breakpoints for the real-time test should be determined by a detailed execution-cost analysis which the compiler could perform (the Fujitsu compiler uses such analyses to make decisions about IF-statements in DO-loops).

4. Implement "detached-singleton" CEs. Because each CE is roughly equivalent to a VAX-8600, an FX/8 with several detached CEs will offer unusually high scalar (plus vector) timesharing performance, concurrently with a cluster of CEs offering high vector/concurrent computing performance.

7.3 Alliant FX/n Performance In November 1985

The Convex and Alliant machines are subject to evolution of both their hardware and their software; benchmark results for these machines are, strictly, valid only for the date of measurement! Alliant made certain changes in the FX/8 hardware and software after September, and asked NRAO to rerun the test procedures to evaluate the effects of these changes. The new tests were run late in November. The only significant differences in the FX/n environment for the November benchmarks were the use of Alliant'a revision "B" CEs and their 8K file system. The 8K file system is different in that the sector sizes are 8192 bytes instead of the normal 512. The 8K file system was used for the storage of AIPS user data only. No changes were made to the AIPS source code which had been used for the September run. However, all source code was recompiled and all load modules were regenerated. The results are shown in Table 3, and lead to the following conclusions:

1. The 8K file system significantly improves I/O performance. The ratio of November to September performance on both the "normal" PFT and "scaled-up" PFT shows little change in CPU times but a general 20% improvement in real times.

2. By ordering the results of the respective PFT tasks in ascending CPU/real-time performance, the ratio of November FX/n performance to September C-1 performance suggests that the 4-6 CE case is roughly comparable to the C-1, particularly for the AP tasks. In the September runs, it seemed that 6-8 CEs were required to match the performance of the C-1.

3. The "scaled-up" PFT problem run was repeated with 6 CEs for the sake of comparison with the September results. Despite Alliant's improved I/O, the real-time superiority of the C-1 still stands.

Table 3

Alliant FX/n Performance Improvements

September 1985:

TASK	IOCNT	FX/1<1> REAL	FX/1<1> CPU	FX/2 REAL	FX/2 CPU	FX/4 REAL	FX/4 CPU	FX/6 REAL	FX/6 CPU	FX/7 REAL	FX/7 CPU	FX/8 REAL	FX/8 CPU			FX/6<5> REAL	FX/6<5> CPU
AIPS	380	<2>	131.2	1549	128.5	1255	128.3	1158	128.1	N/A	N/A	1139	131.1			12513	386.1
CNVRT	25	4	2.1	4	1.6	4	1.4	4	1.4	N/A	N/A	3	1.3			41	9.4
COMB	33	9	6.9	9	6.1	9	5.9	8	5.7	N/A	N/A	8	5.8			110	61.2
SUBIM	19	6	2.8	4	2.5	4	2.4	4	2.3	N/A	N/A	5	2.5			51	9.9
UVSRT	96	25	10.1	28	9.7	25	9.1	25	9.0	N/A	N/A	27	9.4			29	9.2
APCLN	267	307	264.6	201	156.9	151	101.4	134	86.6	N/A	N/A	132	84.5<3>	.89	75.2	4320	1364.2
APRES	82	28	17.6	24	13.7	21	12.0	24	12.2	N/A	N/A	24	12.7<3>	.93	11.9	317	124.6
ASCAL	136	410	394.8	214	202.3	122	110.8	91	80.5	N/A	N/A	78	67.1			160	149.1
MXMAP	96	69	51.8	49	33.0	41	23.8	38	21.9	N/A	N/A	42	23.1<3>	.89	20.6	324	149.2
MXCLN	481	639	573.3	408	336.3	299	230.1	262	186.8	N/A	N/A	234	171.2<3>	.89	152.4	3520	1832.8
UVMAP	151	62	37.6	48	25.2	43	19.0	45	17.8	N/A	N/A	44	18.1<3>	.90	16.3	435	138.7
VM	344	125	64.4	109	49.2	94	41.0	88	41.5	N/A	N/A	84	41.5			1843	569.6
ALL<4>	1730	1684	1426.0	1098	836.5	813	556.9	723	465.7	N/A	N/A	681	437.2			11150	4417.9

	CPU/REAL	CPU/REAL	CPU/REAL	CPU/REAL	CPU/REAL	CPU/REAL	CPU/REAL
AIPS	N/A	N/A	N/A	N/A	N/A	N/A	N/A
CNVRT	.53	.40	.35	.35	N/A	.43	.22
COMB	.77	.68	.66	.71	N/A	.73	.56
SUBIM	.47	.63	.60	.58	N/A	.27	.19
UVSRT	.40	.35	.36	.36	N/A	.35	.32
APCLN	.86	.78	.67	.65	N/A	.64	.32
APRES	.63	.57	.57	.51	N/A	.53	.39
ASCAL	.96	.95	.91	.88	N/A	.86	.93
MXMAP	.75	.67	.58	.58	N/A	.55	.46
MXCLN	.90	.82	.77	.71	N/A	.73	.52
UVMAP	.61	.53	.44	.40	N/A	.41	.32
VM	.52	.45	.44	.47	N/A	.49	.31
ALL	.85	.76	.68	.64	N/A	.64	.40

		FX1/C1		FX2/C1		FX4/C1		FX6/C1		FX7/C1		FX8/C1		FX6/C1	
AIPS	380	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
UVSRT	96	2.78	1.53	3.11	1.47	2.78	1.38	2.78	1.36	N/A	N/A	3.00	1.42	3.63	.63
CNVRT	25	1.33	1.11	1.33	.84	1.33	.74	1.33	.74	N/A	N/A	1.00	.68	1.75	.50
UVMAP	151	2.70	2.05	2.09	1.38	1.87	1.04	1.96	.97	N/A	N/A	1.91	.89	2.50	.98
VM	344	2.78	2.15	2.42	1.50	2.09	1.37	1.96	1.38	N/A	N/A	1.87	1.38	2.75	1.40
APRES	82	2.15	1.74	1.85	1.36	1.62	1.19	1.85	1.21	N/A	N/A	1.85	1.18	2.17	1.15
SUBIM	19	2.00	1.33	1.33	1.19	1.33	1.14	1.33	1.10	N/A	N/A	1.67	1.19	3.40	.76
COMB	33	1.50	1.28	1.50	1.13	1.50	1.09	1.33	1.06	N/A	N/A	1.33	1.07	1.67	1.01
MXMAP	96	2.23	1.90	1.58	1.21	1.32	.88	1.23	.81	N/A	N/A	1.35	.85	1.51	.91
APCLN	267	2.31	2.16	1.51	1.28	1.14	.83	1.01	.71	N/A	N/A	.99	.61	2.41	1.14
MXCLN	481	2.71	2.59	1.73	1.52	1.27	1.04	1.11	.85	N/A	N/A	.99	.69	1.71	1.19
ASCAL	136	3.83	3.81	2.00	1.95	1.14	1.07	.85	.78	N/A	N/A	.73	.65	.72	.69
ALL<4>	1730	2.77	2.60	1.80	1.52	1.33	1.02	1.19	.85	N/A	N/A	1.12	.80	2.07	1.14

Table 3 (continued)

November 1985:

TASK	IOCNT	FX/1<1> REAL	FX/1<1> CPU	FX/2 REAL	FX/2 CPU	FX/4 REAL	FX/4 CPU	FX/6 REAL	FX/6 CPU	FX/7 REAL	FX/7 CPU	FX/8 REAL	FX/8 CPU	FX/6<5> REAL	FX/6<5> CPU
AIPS	380	<2>	126.9	1270	117.8	1110	117.6	961	117.5	962	117.3	904	117.5	8152	387.8
CNVRT	25	4	2.3	3	1.8	3	1.8	2	1.4	2	1.5	3	1.5	24	11.1
COMB	33	8	6.9	8	6.1	7	5.9	7	5.9	8	6.0	7	5.9	72	59.6
SUBIM	19	3	2.8	3	2.5	3	2.3	4	2.3	3	2.3	3	2.6	20	10.6
UVSRT	96	20	10.9	19	9.8	19	9.6	17	9.4	17	9.3	20	10.0	19	9.6
APCLN	267	255	242.4	166	153.2	113	99.6	100	85.7	97	82.6	96	80.4	2575	1446.4
APRES	82	21	16.8	18	13.8	15	11.9	15	12.0	16	12.0	17	11.9	200	125.5
ASCAL	136	395	390.6	198	196.7	111	108.7	110	107.5	110	108.2	74	68.9	216	213.0
MXMAP	96	56	49.3	36	32.4	30	24.1	26	21.9	27	21.0	29	22.1	217	157.5
MXCLN	481	515	500.2	349	333.2	245	228.7	208	190.1	221	194.7	179	161.7	2342	1840.5
UVMAP	151	46	37.0	35	26.0	31	20.1	31	19.1	30	19.0	31	18.9	296	156.7
VM	344	81	61.9	57	45.0	55	40.9	62	43.8	65	43.4	53	39.3	1069	625.3
ALL<4>	1730	1404	1321.1	892	820.5	632	553.3	582	499.1	593	500.0	512	423.2	7050	4655.8
		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL		CPU/REAL	
AIPS		N/A		N/A		N/A		N/A		N/A		N/A		N/A	
CNVRT		.58		.60		.60		.70		.75		.50		.46	
COMB		.86		.76		.84		.84		.75		.84		.83	
SUBIM		.93		.83		.77		.57		.77		.87		.53	
UVSRT		.55		.52		.51		.55		.55		.50		.51	
APCLN		.95		.92		.88		.86		.85		.84		.56	
APRES		.80		.77		.79		.80		.75		.70		.63	
ASCAL		.99		.99		.98		.98		.98		.93		.99	
MXMAP		.88		.90		.80		.84		.78		.76		.73	
MXCLN		.97		.95		.93		.91		.92		.90		.79	
UVMAP		.80		.74		.65		.62		.63		.61		.53	
VM		.76		.79		.74		.71		.67		.74		.58	
ALL		.94		.92		.88		.86		.84		.83		.66	
		FX1/C1		FX2/C1		FX4/C1		FX6/C1		FX7/C1		FX8/C1		FX6/C1	
AIPS	380	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
UVSRT	96	2.22	1.65	2.11	1.48	2.11	1.45	1.89	1.42	1.89	1.41	2.22	1.52	2.38	1.43
CNVRT	25	1.33	1.21	1.00	.95	1.00	.95	.67	.74	.67	.79	1.00	.79	1.00	.50
UVMAP	151	2.00	2.02	1.52	1.42	1.35	1.10	1.35	1.04	1.30	1.04	1.35	1.03	1.70	1.11
VM	344	1.80	2.06	1.27	1.50	1.22	1.36	1.38	1.46	1.44	1.45	1.18	1.31	1.59	1.54
APRES	82	1.62	1.66	1.38	1.37	1.15	1.18	1.15	1.19	1.23	1.19	1.31	1.18	1.37	1.16
SUBIM	19	1.00	1.33	1.00	1.19	1.00	1.10	1.33	1.10	1.00	1.10	1.00	1.24	1.33	.82
COMB	33	1.33	1.28	1.33	1.13	1.17	1.09	1.17	1.09	1.33	1.11	1.17	1.09	1.09	.98
MXMAP	96	1.81	1.81	1.16	1.19	.97	.89	.84	.81	.87	.77	.94	.81	1.01	.96
APCLN	267	1.92	1.98	1.25	1.25	.85	.81	.75	.70	.73	.68	.72	.66	1.43	1.21
MXCLN	481	2.18	2.26	1.48	1.51	1.04	1.03	.88	.86	.94	.88	.76	.73	1.14	1.20
ASCAL	136	3.69	3.77	1.85	1.90	1.04	1.05	1.03	1.04	1.03	1.04	.69	.66	.98	.98
ALL<4>	1730	2.31	2.41	1.46	1.50	1.04	1.01	.96	.91	.97	.91	.84	.77	1.31	1.20

Table 3 (conclusion)

		FX/1<1> NOV/SEP		FX/2 NOV/SEP		FX/4 NOV/SEP		FX/6 NOV/SEP		FX/7 NOV/SEP		FX/8 NOV/SEP		FX/6<5> NOV/SEP	
AIPS	380	<2>	.96	.82	.92	.88	.92	.83	.92	.84	.89	.79	.90	.65	1.00
CNVRT	25	1.00	1.10	.75	1.13	.75	1.29	.50	1.00	N/A	N/A	1.00	1.15	.59	1.18
COMB	33	.89	1.00	.89	1.00	.78	1.00	.88	1.04	N/A	N/A	.88	1.02	.65	.97
SUBIM	19	.50	1.00	.75	1.00	.75	.96	1.00	1.00	N/A	N/A	.60	1.04	.39	1.07
UVSRT	96	.80	1.08	.68	1.01	.76	1.05	.68	1.04	N/A	N/A	.74	1.06	.66	.97
APCLN	267	.83	.92	.83	.98	.75	.98	.75	.99	N/A	N/A	.73	1.07	.60	1.06
APRES	82	.75	.95	.75	1.01	.71	.99	.63	.98	N/A	N/A	.88	1.02	.63	1.01
ASCAL	136	.96	1.00	.93	.97	.91	.98	1.21	1.34	N/A	N/A	.95	1.03	1.35	1.43
MXMAP	96	.95	.95	.73	.98	.73	1.01	.68	1.00	N/A	N/A	.69	1.07	.67	1.06
MXCLN	481	.81	.87	.86	.99	.73	.99	.79	1.02	N/A	N/A	.76	1.06	.67	1.00
UVMAP	151	.74	.98	.73	1.03	.72	1.06	.69	1.07	N/A	N/A	.70	1.16	.68	1.13
VM	344	.65	.96	.52	.91	.59	1.00	.70	1.06	N/A	N/A	.63	.95	.58	1.10
ALL<4>	1730	.83	.93	.81	.98	.78	.99	.80	1.07	N/A	N/A	.75	.97	.63	1.05

<1> compiled with concurrency disabled (trying to simulate an FX/1 environment on an FX/8)
<2> tasks initiated standalone to insure a 1 CE environment; AIPS real time is not meaningful
<3> from an earlier run when routines QMINV and QMAXV were not vector-concurrent; these tasks stand to gain approximately 10% speedup as shown (8 CEs not available for final run)
<4> tasks only (i.e., does not include AIPS itself); AIPS and its dependencies compiled with concurrency disabled
<5> scaled up problem (i.e., CELLSIZE=2 vs 8, IMSIZE=1024 vs 256, GAIN=0.01 vs .1, NITER=4000 vs 2000)

7.4 Measurements Of "Alliant Concurrency"

The first thing to note is that NRAO's objective was procurement, not parallelism research. Measurements of parallelism were made in order to gain insight into the behavior of the FX/n, the first concurrent computer ever tested by NRAO. The measurements are presented in Table 3. The discussions of "Alliant Concurrency" in this section will refer to the data presented in the "September 1985" section of Table 3. Note that the FX/n is still gaining in performance for N up to 8 in several programs. The data suggest that 4-6 CEs is a good compromise configuration; this would make especially good sense if extra CEs could be detached for timesharing support.

The Alliant CEs are capable of surprising performance utilizing only their scalar concurrency capability. For example, vector sines and cosines are not, in fact, computed in the vector registers. Instead they are computed in the scalar floating point hardware. With 8 CEs the resulting speed is impressive: about 4.5 microseconds per CE divided by 8 implies about 0.5 microsec/result, more than twice as fast as the AP-120B (1.3 microsec/result). Another example: In Table 3 the September CPU time for MXCLN with 8 CEs (the "FX/8" column) using the vectorized CLEAN algorithm is 171 seconds (ignoring the probable speedup if QMINV and QMAXV vectorize). An experimental version of the CLEAN algorithm consisting of simple concurrent scalar code in 8 CEs executed in September in about 200 seconds. Thus, the concurrent-vector technique was only moderately faster than concurrent-scalar in this case! The authors infer that concurrent scalar compilation is a good choice for loops which are difficult to vectorize.

The data in Table 3 include performance measurements for 1, 2, 4, 6 and 8 CEs. The following simple model can be fitted to the data:

CPU_time = (scalar_time + concurrent_time / #CE)

As an example, this formula was solved for the "scalar_time" and "concurrent_time" variables using the data for 2 CEs and 6 CEs for the three tasks ASCAL, MX-clean and VM; the results are tabulated in Table 4 and plotted in Figure 1. The "fraction concurrent" column indicates the fraction of each algorithm's work which was capable of executing in parallel on the FX/8 (for ASCAL, 0.95 is 363 divided by 363+21). The "infinity speedup" indicates how much faster than 1 CE the Alliant architecture would be if an infinite number of CEs were available (for ASCAL, 18.3 is 363+21 divided by 21, i.e. concurrent_time asymtotically zero). We see that ASCAL is the most highly vectorized/parallelized AIPS task (95% concurrent). MX-clean is doing well (80%), but probably could be improved. It is clear that VM needs work---only 38% of its computing was able to exploit multiple CEs in the Alliant. Mainly, the version of VM used in this work needed vectorization directives inserted into its source in various places. Also, certain equivalenced variables were inhibiting vectorization; these problems were subsequently fixed in the Cray X-MP implementation of VM.

Table 4

Speedups Limited by Scalar Code ("Amdahl's Law")

task	scalar (secs)	concurrent (secs)	fraction concurrent	infinity speedup
ASCAL	21	363	0.95	18.3
MX-clean	112	447	0.80	5.0
VM	38	23	0.38	1.6

Figure 1