Tape and TV Performance in AIPS

Eric W. Greisen

August 26, 1992

1 Introduction

The NRAO will be phasing out its Convex C-1 computers beginning in the next calendar year. It would be desirable, according to some people, to turn them completely off on January 1, 1993. The principal losses in so doing would be the four high-speed, half-inch reel tape drives on each C-1 and the IVAS and IIS Model 70 TV display devices supported by the C-1's. It is therefore necessary for us to evaluate the extent of, and to attempt to minimize, these losses.

In $\mathcal{AIPS}$ Memo 80, I addressed the subject of the implementation of remote tape devices in $\mathcal{AIPS}$, including measurements of performance for various configurations of the Berkeley sockets used to implement them. Additional questions that arise with tapes are whether $\mathcal{AIPS}$ even works with particular tape devices and, if so, how well. This memo presents some measurements of performance on digital audio tapes, Exabyte tapes, and 6250-bpi, half-inch reel tapes on a Sun IPX ("primate"), an IBM RS 6000/530 ("lemur"), and a Convex C-1 ("nrao1"). The Exabyte on lemur is an 8200 model, while the one on primate is a dual-density model used in 8200 mode.

After a very useful discussion with Richard Gooch of the Australia Telescope, I began a number of modifications to the $\mathcal{AIPS}$ "television" display driver for workstations (`XAS`). This memo also addresses briefly the nature of those modifications and presents some measurements of the changes in performance. In addition, the results are compared to performances on the IIS Model 70 and IVAS displays on nrao1.

2 Tape performance

A number of test programs were run on the various computers and tape drives. These were primarily `FITTP` to write FITS-format data to tape, `PRTTP` to read each record of the tape including parsing the headers and printing summaries, and `AVTP` to advance to the end of tape reading one record followed by an advance-file for each file on the tape.

The results given in the Table 1 below are not surprising. Real half-inch tape drives are faster, but, of course, hold very little data by modern standards. Those on lemur are faster than those on the older nrao1. Exabytes are faster than DATs by a modest margin when the data files are around 20 Mbytes or more, but DATs are much faster at handling end-of-file marks. Thus DATs are to be preferred for files around 3 Mbytes. It should be possible to quantify this by assuming that

$$T_{real} - T_{cpu} = N_{files} X + M_{bytes} Y$$

where N_{files} is the number of files processed and M_{bytes} is the number of Megabytes of data. The fit to this model is good in most cases and the results are presented in Table 2. If the numbers are to be believed, Exabytes have a heavy overhead per file for writing, but run about 1.5 times faster per Megabyte than DATs. They have more similar speeds when reading.

Table 1. Tape Operation Times (seconds)

Function	Size		Computer	Exabyte		DAT		1/2-inch reel	
	N_{files}	M_{bytes}		T_{cpu}	T_{real}	T_{cpu}	T_{real}	T_{cpu}	T_{real}
FITTP	49 uv	150M	primate	337.7	3147	338.8	2010	-	-
FITTP	16 uv	438M	primate	671.4	2613	669.5	2823	-	-
FITTP	18 uv	484M	primate	742.7	2917	736.9	3139	-	-
FITTP	14 uv	333M	lemur	341.4	2136	342.5	2206	-	-
FITTP	7 uv	148M	lemur	153.6	1040	153.4	988	157.1	448
FITTP	23 uv	4.5M	lemur	50.0	1214	50.3	401	50.2	156
FITTP	3 uv	118M	lemur	118.7	705	118.6	719	114.6	296
FITTP	1 uv	0.3M	nrao1	-	-	-	-	5.3	11
FITTP	23 uv	3.4M	nrao1	-	-	-	-	202.0	395
FITTP	48 uv	151M	nrao1	-	-	-	-	920.9	1854
FITTP	1 uv	28M	nrao1	-	-	-	-	103.4	193
PRTTP	97 uv	1405M	primate	244.9	6615	249.5	8281	-	-
PRTTP	7 uv	148M	primate	19.4	704	20.4	871	-	-
PRTTP	97 uv	1405M	lemur	174.8	6932	176.0	8272	-	-
PRTTP	7 uv	148M	lemur	13.4	748	14.2	871	15.3	230
PRTTP	23 uv	4.5M	lemur	32.0	192	31.7	51	31.6	50
PRTTP	3 uv	118M	lemur	8.7	587	8.5	697	8.9	182
PRTTP	47 uv	142M	nrao1	-	-	-	-	7.6	66
PRTTP	1 uv	28M	nrao1	-	-	-	-	7.6	66
TPHEAD	1 uv		primate		33		11	-	-
TPHEAD	1 uv		lemur		16		14		8
AVTP	49 uv	150M	primate	0.3	567	0.3	389	-	-
AVTP	65 uv	588M	primate	0.3	941	0.3	650	-	-
AVTP	97 uv	1405M	primate	0.6	1677	0.5	1133	-	-
AVTP	7 uv	148M	primate	0.2	174	0.2	100	-	-
AVTP	83 uv	1027M	lemur	0.2	1070	0.3	964	-	-
AVTP	7 uv	148M	lemur	0.1	150	0.1	96	0.1	220
AVTP	23 uv	4.5M	lemur	0.1	183	0.2	129	0.1	12
AVTP	3 uv	118M	lemur	0.1	110	0.1	44	0.1	178
REWIND	97 uv	1405M	primate		119		57	-	-
REWIND	7 uv	148M	primate		40		15	-	-
REWIND	97 uv	1405M	lemur		109		78	-	-
REWIND	7 uv	148M	lemur		36		35		90
REWIND	23 uv	4.5M	lemur		27		6		8
REWIND	3 uv	118M	lemur		34		29		75
DISMOUNT	49 uv	150M	primate		56		42	-	-
DISMOUNT	7 uv	148M	primate		59		39	-	-

Table 2. Apparent Tape Rates

Computer	Tape	X (sec/file)			Y (sec/Mbyte)		
		FITTP	AVTP	PRTTP	FITTP	AVTP	PRTTP
nrao1	reel	8	-	4.0	3.50	-	2.0
primate	DAT	21	6.9	-1.0	4.15	0.35	5.8
primate	Exabyte	49	9.5	-7.0	2.65	0.60	4.8
lemur	reel	4.2	0.2	0.5	1.40	1.50	1.45
lemur	DAT	14	5.6	-0.3	4.70	0.20	5.84
lemur	Exabyte	50	7.8	6.0	3.80	0.73	4.75

3 Changes to XAS and Performance

A number of changes have been made in the 15OCT92 version of the *AIPS* television driver XAS. First, the DISPLAY variable was changed from *host*:0 to simply :0. This should prompt the X server to use Unix sockets rather than Internet sockets, with some improvement in performance. Second, the "blit" of the image from XAS's memory to the display was changed to be as large as possible on each display update. Previously, only a row at a time was blitted when the image was zoomed and/or contained graphics overlays. Third, the XAS memory was changed to use, optionally, the X extension called "shared memory." This greatly improves blit speed after an initial overhead to synchronize the memories. Fourth, the application code was provided with the option to ask XAS to delay updating the display until instructed to do so. This allows multiple graphics planes to be turned on with a single screen update, a full image to be loaded with a single blit to the display rather than one blit per row, multiple line segments of a plot to be drawn with a single blit to the display rather than very many small blits, and so forth. This option, implemented with subroutine YHOLD, is dangerous in that it requires considerable care on the part of the application programmer to make certain the the display is brought up to date whenever required. As some protection against programmer error, subroutine TVCLOS forces synchronization. Also the new XAS allows the user to set (via his or her .Xdefaults file) a maximum number of commands to be done asynchronously before XAS itself forces an update of the screen.

The two tables below list some times to complete and some frames rates for various TV functions using nrao1 for the hardware TV devices (IIS and IVAS) and primate for various versions of XAS.

Table 3. TV Operation Times (seconds) (smaller numbers are better)

Function	15APR92	15OCT92	15OCT92	15OCT92	15OCT92	IIS	IVAS
Computer	primate	primate	primate	primate	primate	nrao1	nrao1
Asynchronous?	No	No	Yes	No	Yes	na	na
Shared memory?	No	No	No	Yes	Yes	na	na
25 TVINITs	150	125	89	73	69	30	162
25 TVLODs (256)	53	58	41	71	40	92	105
25 TVLABELs	323	267	150	344	162	64	94
CNTR (real)	70	64	33	98	36	28	36
CNTR (cpu)	13.0	16.3	16.0	17.6	16.5	14.3	12.5

Table 4. TV Maximum Frames / second (larger numbers are better)

Function	Size		15APR92	15OCT92	15OCT92	IIS	IVAS
Computer			primate	primate	primate	nrao1	nrao1
Shared memory?			No	No	Yes	na	na
TVblink	518	518	3.26	4.70	13.9	7.5	—
TVblink	1142	800	1.00	1.54	10.0	—	3.9
TVmovie no zoom	258	198	13.50	21.00	47.0	—	—
TVmovie 2x zoom	570	396	2.07	3.36	6.8	7.5	—
TVmovie 3x zoom	855	594	0.98	1.72	3.8	—	—
TVmovie 4x zoom	1140	792	0.62	1.05	3.4	—	6.35

The values in the first table may be understood after some reflection. The overhead of synchronizing shared memory to display memory is not trivial. Therefore, shared memory can be very slow when the displays are done a small amount at a time, as is usually required in $\mathcal{AIPS}$, unless the screen updates are combined via the asynchronous option. In fact, for the image drawing functions in the first table, the use of the asynchronous option is very much more important than the shared memory option and regular memory is preferable to shared in two of the four tests.

The second table was prepared with special versions of `TVBLNK` and `TVMOVI` which were altered to run at maximum rates (no calls to `ZDELAY`) and to report the frame rates on button pushes. There is no way that the asynchronous option may be used in these algorithms; they are simply a measure of how quickly can we blit portions of the image memories to the display (with zoom computations where needed). Clearly shared memory is a big winner in these algorithms, pushing the screen hardware update rates in the fastest case.

I do not understand why the IIS frame rates are one-fourth of the screen refresh rate. There was a background `MX` running during all of the nrao1 tests. However, numerous frame rate measurements gave consistent results, suggesting that `MX` was not to blame. Ignoring this (small) uncertainty, it is clear that the new `XAS` is quite competitive with the old hardware TVs for these standard functions. Of course, `XAS` cannot display true-color images, nor can it do our hue-intensity algorithm.