22 July 71

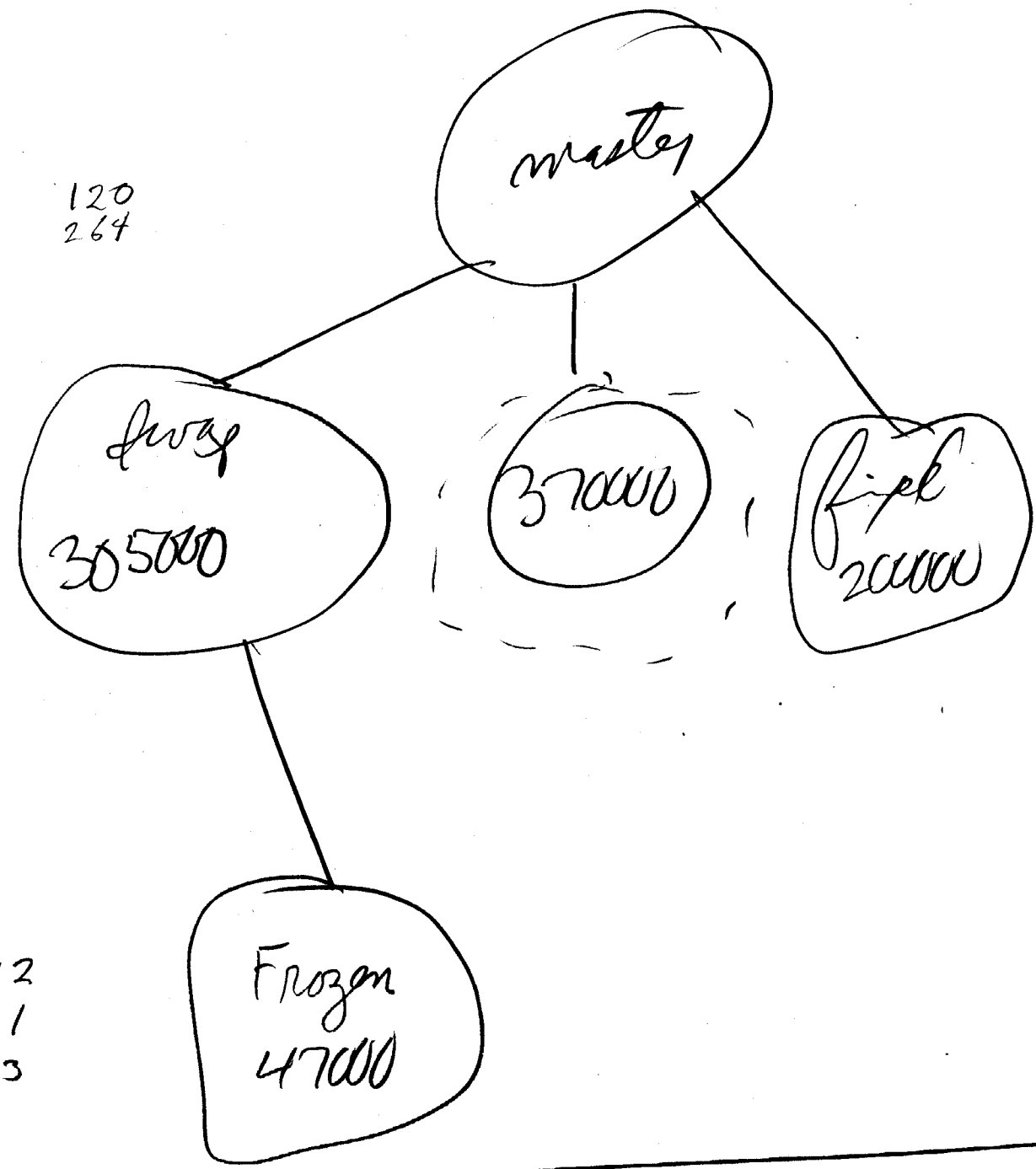
Where ECS space is going. In May, we investigated ECS space consumption. Page A gives the ~~MACRO~~ big picture in terms of what ABR have how much space allocated to them.

The 370K below the "master" is system overhead, not all of which is accounted for.

Page B is an analysis of why the fixed ECS requirements for a provide are so large (page C is ~~on~~ previous version of the same thing). It showed that the ~10K (we didn't verify that the total on the page is the real total, but it is close (~~unlikely~~ that 400 escaped our audit)) is evenly divided amongst the subprocesses. Scratch files are the big expense.

The builder + all its overhead have been eliminated now.

A



120
264

1042
81
1103

11012_{10}

5/19/71 Fixed ECS overhead in user process

SUBPROCESS OVERHEAD

Subprocess	# log. reqs	core reqs	dict	scratch	subpgd	other	total
USRDSK	6	43	30	1077	11	10	1271 ✓
FRETRW	4	17	25	516	11	10	650 ✓
DIRSYS	3	24	13	230	11	10	340 ✓
NULLSUB	2	4?	4	101	11	10	57
BEADS	5	24	41	421	11	10	613 ✓
FAKEG	5	24	10	147	11	10	257 ✓
BEADG	5	24	13	206	11	10	324 ✓
TLINE	2	12	14	252	11	10	347 ✓
CMMD	5	40	104	502	11	10	1016 ✓
BLDR	5	62	111	240	11	264? *	1064 ×
ROOT	3	10	10	0	11	10?	66
TOTAL	55 ✓	352 ✓	421 ✓	4267 ✓	143	404 ✓	6735 ✓
	(3N+1 per)		(2N+3 per)				
	222	352	1103	4267 ✓	143 ✓	404 ✓	6735 ✓

PROCESS DESCRIPTOR

167	basic descriptor
7	QUEUE SIZE+1
12?	Full C-list Table (2*max depth tree)
74	stack (2* # entries)
2	?
306	

* 10 scratch file overhead
77 a c-list (size 36)
155? another level file
unknown C-list

7243

(C)

5/18/71

	Humap entries	comp map size	clist size	sws z	
USRDSU	6 [23]	43	30 [63]	1077	1250 +10
Frotrw	4 [15]	17	25 [58]	516	627 ✓
Dirsys	3 [12]	24	13 [31]	230	317 ✓
Nullsub	2 [7]	62→74 *	4 [13]	102→70 *	206 ✓
Reads	5 [20]	24	41 [105]	421	572 ✓
Faney	5 [20]	24	10 [23]	147	236
Beady	5 [20]	24	13 [31]	206	303 ✓
TUNE	2 [7]	12	14 [33]	252	326
CMMD	5 [20]	40	104 [213]	502	+27 775 ✓
Bldr	5 [20]	62	111 [225]	240	to other files: 36 (clist) 145 (file) to another clist 567 ✓ 45 ✓
Root	3 [12]	10	10 [23]	0	
	55 ✓	430 ✓	420 ✓	4361 ✓	

* 3N+1 — * 2N+3 — + file overhead

for SP devices: 207+13 — 1042+41

140 ✓ 222 ✓ 430 ✓ 1103 ✓ 4...

6336
143
6501

6926

+ file overhead on scratch files
+ process overhead
+ 77 clist for bldr
+ 145+ file for bldr

167	P.LOCALC
7	+ QBUFF+1 ←
12-14	+2*depth+1
74	+2*stack
2	+2
306	

(* improvements not included

6745
130 + file overhead

7075
306
7403