NASA Technical Memorandum 105200

Cray Performance Data From Five Benchmarks

James A. Pennline
Lewis Research Center
Cleveland, Ohio

September 1991

NASA

CRAY PERFORMANCE DATA FROM FIVE BENCHMARKS

James A. Pennline
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 4413

SUMMARY

The five benchmark programs discussed in TM 88956, February 1987, were since then run on the CRAY X-MP/24 under different operating systems and compilers. Performance data is reported for runs under early versions of COS and CFT up through runs under current versions of UNICOS and CFT77. The most recent data includes a system configuration for a X-MP hardware upgrade. Performance figures for the Y-MP are also shown for comparison. Differences in the figures are analyzed and discussed.

INTRODUCTION

One of the objectives for collecting and comparing the five benchmarks reported in TM 88956 was to run them whenever changes were made to the operating system or hardware to observe the effects on the performance data. The following data was collected at various stages of the CRAY configuration. The stages include system configurations which evolved from initial versions of COS and CFT to present versions of UNICOS and CFT77. The results of the runs of the benchmarks with selected upgrades/changes in the operating system and compiler are presented in tabular form. The effect of the change from one operating system or compiler to the next in the performance can be seen by reading from left to right across the tables.

The tables of results shown next are for system configurations in the period March 1987 to April 1989. Following the tables, some of the significant differences are noted and discussed.

Program	COS1.14BF4 CFT1.14	COS1.15BF2 CFT1.15BF2	COS1.15BF2 CFT77(1.3)	COS1.16BF2 CFT1.15BF2	UNICOS4.0 CFT77(2.0)

NAS Kernels

MXM	136	136	178	137	173
CFFT2D	51	49	43	49	55
CHOLSKY	53	56	42	57	58
BTRIX	80	74	34	74	94
GMTRY	70	73	7	81	74
EMIT	82	82	81	86	89
VPENTA	41	42	42	42	41
TOTAL	65	65	30	66	71

Sandia SPEED

Kernel 1	23	23	25	23	33
2	11	63	75	63	71
3	39	37	52	39	51
4	10	10	13	11	12
5	8	8	8	8	8
TOTAL	13	16	18	16	19

WHETSTONE

1 meg instr(s)	25	25	32	25	31

THE ARGONNE PROGRAMS

COS1.14BF4 CFT1.14	COS1.15BF2 CFT1.15BF2	COS1.15BF2 CFT77(1.3)	COS1.16BF2 CFT1.15BF2	UNICOS4.0 CFT77(2.0)

LINPACK

ORD 100	22	25	36	25	28

Better LU decomposition ORD 100

UD 1	32	31	53	32	55
2	42	42	61	43	63
4	50	50	62	50	65
8	57	57	62	57	63
16	57	57	62	58	62

Better LU decomposition ORD 300

UD 1	68	68	100	69	99
2	88	88	120	88	119
4	99	100	123	100	123
8	115	116	125	116	121
16	117	117	125	118	120

UD - Unrolled depth

Vector Loops

COS1.14BF4 CFT1.14	COS1.15BF2 CFT1.15BF2	COS1.15BF2 CFT77(1.3)	COS1.16BF2 CFT1.15BF2	UNICOS4.0 CFT77(2.0)

Loop

1	n	n		n	n
2	y	y		y	n
3	y	y		y	y
4	n	n		n	y
5	y	y		y	y
6	y	y	NO	y	y
7	n	y		y	y
8	n	n		n	n
9	y	y	DATA	y	y
10	n	n		n	y
11	y	y		y	y
12	y	y		y	y
13	y	y		y	y
14	y	y		y	y
15	n	y		y	y
16	n	n		n	n
17	y	y		y	y

1 Statements in wrong order
2 Dependency needing a temporary
3 Loop with unnecessary scalar store
4 Loop with ambiguous scalar temporary
5 Loop with subscript that may seem ambiguous
6 Recursive loop that really isn't
7 Loop with possible ambiguity because of scalar store
8 Loop that is partially recursive
9 Loop with unnecessary array store
10 Loop with independent conditional
11 Loop with noninteger addressing
12 Simple loop with dependent conditional
13 Complex loop with dependent conditional
14 Loop with singularity handling
15 Loop with simple gather/scatter subscripting
16 Loop with multiple dimension recursion
17 Loop with multiple dimension ambiguous subscripts

Livermore Loops

Kernel	COS1.14BF4 CFT1.14	COS1.15BF2 CFT1.15BF2	COS1.15BF2 CFT77(1.3)	COS1.16BF2 CFT1.15BF2	UNICOS4.0 CFT77(2.0)
1	152	152	165	152	163
2	26	27	39	28	45
3	135	135	155	135	143
4	44	39	63	40	62
5	6	6	14	6	14
6	13	12	15	12	15
7	171	171	187	171	187
8	113	118	140	118	148
9	144	144	161	145	155
10	65	69	71	69	41
11	8	8	13	8	12
12	71	71	82	72	83
13	4	4	6	5	6
14	11	11	21	13	14
15	5	5	6	5	6
16	3	3	7	4	7
17	9	11	11	12	11
18	112	111	116	112	128
19	7	7	15	8	15
20	12	12	13	12	13
21	29	28	64	29	62
22	66	66	67	66	68
23	13	13	14	14	13
24	2	2	3	2	3

DISCUSSION OF SOME SIGNIFICANT CHANGES

(March 1987 to April 1989)

CFT1.14 to CFT 1.15

Note the increase in MFLOPS from 11 to 62 in kernel 2 of the Sandia program . This happened basically because a loop in the program which did not vectorize under CFT1.14 vectorized under CFT1.15. It is the inner loop of the following multiple loop.

```
      DO 150 I = 1, K2
         TEMP = FLOAT(I)/DENOM
         XL = 0.
         DO 140 L = 1, NEQN2
            XL = XL + 1.
            YH(L,I) = TEMP + XL / XEQN2
140      CONTINUE
150   CONTINUE
```

The vector loop program from the Argonne collection shows that loop 7,

```
      DO 70 I = 1, NO7-1
         J = I + 1
         VO7A(I) = VO7A(J)
 70   CONTINUE,
```

which tests the compiler's ability to handle a possible ambiguity because of a scalar store, did not vectorize under 1.14 but did under 1.15. Thus the compiler has gotten slightly smarter in handling these types of loops. The reason for saying slightly is that apparently the compiler can vectorize a loop similar to the one above so long as a scalar such as XL is incremented in a regular fashion. We conducted a test on loops of the following form.

```
      DO 10 L = 1, N
         XL = FCN(XL)
         YL = XL
 10   CONTINUE
```

It will vectorize if FCN(XL) = XL + K where K is a fixed integer. Otherwise, the compiler refuses to vectorize for reason that a value carried around the loop is not incremented in a regular fashion.

Most of the rest of the comparison between CFT1.14 and CFT 1.15 shows both slight increases and slight decreases in MFLOPS.

CFT1.15 to CFT77 (version 1.3)

The third column shows data for CFT77 version 1.3. The numbers in this column reflect some problems with early versions of CFT77. Notice the NAS kernels. Although the MFLOP rate for MXM went up, others went down.

Significantly, the rates for BTRIX and GMTRY were reduced by a half and a factor of 10 respectively. The problem with BTRIX is that there were three

loops which did not vectorize directly in CFT77 but did vectorize directly under CFT1.14 and CFT1.15. One of the loops is the following.

```
      DO 100 J = JS, JE
      IF (J.EQ.JS) GO TO 4
         DO 3 M = 1,5
           DO 3 N = 1,5
             DO 3 L = LS, LE
               B(M,N,J,L) = B(M,N,J,L) - A(M,1,J,L)*B(1,N,J-1,L)
                 - A(M,2,J,L)*B(2,N,J-1,L) - A(M,3,J,L)*B(3,N,J-1,L)
                 - A(M,4,J,L)*B(4,N,J-1,L) - A(M,5,J,L)*B(5,N,J-1,L)
       3     CONTINUE
       4 CONTINUE
                                .
                                .
                                .
     100 CONTINUE
```

The other two loops are similar. The inner most loop above did vectorize but only after a significant amount of code was generated to test it first. Thus the loops referred to above conditionally vectorized. The problem with GMTRY was a triple nested loop which did not vectorize, i.e.,

```
      DO 8 I = 1, MATDIM
        RMATRX(I,1) = 1. / RMATRX(I,1)
        DO 8 J = I+1, MATDIM
          RMATRX(J,I) = RMATRX(J,I) * RMATRX(I,1)
          DO 8 K = I+1, MATDIM
            RMATRX(J,K) = RMATRX(J,K) - RMATRX(J,I) * RMATRX(I,K)
    8 CONTINUE
```

A check was made for dependency and the inner loop on K did not vectorize.

The rates in the rest of the benchmark programs either remained the same or increased. Like MXM in the NAS kernels, kernel 21 in the Livermore Loops, which is a matrix-matrix product increased significantly. Kernel 3 had the biggest increase among the Sandia SPEED kernels. It does a forward and backward substitution excerpt from a linear equation solver with pivoting. A better handling of recursion is reflected in the roughly doubling of the rates for kernels 5 and 19 in the Livermore Loops. Also, there is a roughly doubling of performance in kernels 14 and 16, which do a particle in cell excerpt and a Monte Carlo search respectively.

COS1.15 (CFT1.15) to COS1.16 (CFT1.15)

There were no significant changes between these two configurations aside from some very slight increases in the rates.

CFT77 (version 2.0) under UNICOS 4.0

This was our first look at the MFLOP rates under a UNICOS system. Starting with the NAS kernels, we see that the conditional vectorization and no vectorization problems encountered with the BTRIX and GMTRY kernels, respectively, when compiled under CFT77 (version 1.3) have been corrected

under CFT77 (version 2.0). In fact, the rates are even slightly higher than previous CFT and COS configurations. Except for the MXM kernel, the rates for the other kernels are the same or slightly higher than CFT77 (1.3) and previous CFT version rates. In the case of MXM, the rate is higher than all previous CFT version rates but slightly lower than the CFT77 (1.3) rate. This can be due to differences in the UNICOS versus COS operating systems.

The rates for the Sandia SPEED kernels are essentially the same or slightly higher than previous CFT version rates. Comparing CFT77 (1.3) to CFT77 (2.0) kernel 2 had a slightly lower rate under version 2.0 which again may be due to differences in the operating system.

The Whetstone rate is essentially the same as the rate for CFT77 (1.3) under COS and higher than all previous CFT rates under COS.

Next to note are the Argonne Programs. The LINPACK rate is slightly higher than all previous CFT version rates. It's not as high as the CFT77 (1.3) rate under COS1.15 but again this can be due to differences in instruction sequence, scheduling, etc. between UNICOS4.0 and COS1.15. Also remember that LINPACK is not well suited to show the kind of performance a vector machine can give. It uses the BLAS package, and we ran the code as is without inlining subroutines. Note the rates for the better LU decomposition which uses matrix-vector techniques and is better suited to make use of vectorizaton capabilities. The rates are improved over rates from previous CFT versions and are essentially the same as rates obtained from CFT77 (1.3) under COS1.15. The Vector Loops program shows a few changes. Two of the loops which did not vectorize under CFT now vectorize under CFT77 (2.0). One of them is loop 4 which is the following:

```
      T = 0.
      DO 40, I = 1, N04
         S = V99A(I)*V99B(I)
         V04A(I) = S+T
         T = S
   40 CONTINUE
```

This loop is identified as a loop with an ambiguous scalar temporary in the Argonne program. In the CRAY CFT Optimization Guide SG 0115, 1/88, this specific loop is exemplified as one which will not vectorize because the scalar temporary T is not defined in the loop before it is used on the right hand side of an equal sign. The Optimization Guide shows a modified version which will vectorize under CFT. With CFT77 the above loop now vectorizes without modification. (However, we found a slight bug in that if a write statement to write out V04A is inserted before the do, CFT77 then refuses to vectorize it for with the explanation that values carried around the loop are not incremented in a regular fashion.) The other loop which now vectorizes is loop 10, namely

```
      T = 1.
      DO 100, I = 1, N10
         IF (V99C(I).GE.T) THEN
            X = V99A(I)*V99B(I)+3.1
            Y = V99A(I)+V99B(I)*2.9
            V10A(I) = SQRT(X**2*Y)
          ENDIF
100   CONTINUE
```

which is defined as a loop with independent conditional. A more interesting observation is that loop 2 which vectorized under CFT does not under CFT77. Loop 2 is the following

```
      IF (ABS(V02B(2).GT.MAXUM) OP02 = .NOT.OP02
      IF (OP02.EQV.ADD) THEN
         DO 20, I=1,N02-1
            V02A(I) = V99A(I)
            V02B(I) = V02B(I)+V02A(I+1)
20       CONTINUE
         ELSE
         DO 21, I=1,N02-1
            V02A(I) = V99A(I)
            V02B(I) = V02B(I)-V02A(I+1)
21       CONTINUE
       ENDIF
```

This loop is defined as a dependency needing a temporary in the Argonne program. In the CRAY CFT optimization guide, a similar loop is defined as an SPI conflict. What this means is that the array V02A has two appearances, a key definition and another appearance in a subsequent area with an index incremented by 1. The conflict can be eliminated by changing the order of the assignment statements and enabling vectorization. Apparently CFT did this automatically but CFT77 does not.

Last to note are some of the significant differences in the 24 kernels of the Livermore Loops. In particular, there is roughly a doubling of the rates for loops 5, 11, and 19 compared to rates for previous CFT versions. Since these involve recursion, this exemplifies a better handling of certain types of nonvectorizable recursive loops. Loop 3 had a slightly higher rate than previous CFT rates but a slightly lower rate than the CFT77 (1.3) rate. We are unsure if this is due to operating system or compiler differences. Loop 18 shows an improved rate over CFT and CFT77 (1.3) rates, which may be due to the elimination of conditional vectorization. Most significant is the drop in the rate for loop 10. Some detailed investigation had to be done to determine why this happened, and an explanation will be given under the discussion of performance for UNICOS 5.0 and CFT77 (3.1) where the rate dropped even lower.

The next set of performance figures shows the MFLOP rates for the same benchmark programs as upgrades were made to UNICOS and CFT77. The next to the last column shows rates after a hardware upgrade when the CRAY X-MP 24 model 128 (2 cpu(s) and 4M words of memory) was replaced by a CRAY X-MP 28 model 426 (2 cpu(s) and 8M words of memory). The CPU clock cycle went from 9.5 to 8.5 ns and the chip technology changed from ECL base memory to CMOS base memory. We also included in the last column performance figures for the Y-MP.

				X-MP SWAP	Y-MP
Program	UNICOS4.0 CFT77(2.0)	UNICOS5.0 CFT77(3.1)	UNICOS5.1.8 CFT77 Ver 3.1.2.8	UNICOS5.1.10 CFT77 Ver 3.1.2.8	UNICO5.1.10 CFT77 Ver 3.1.2.8

NAS Kernels

MXM	173	182	182	202	284
CFFT2D	55	55	55	36	73
CHOLSKY	58	58	59	63	89
BTRIX	94	99	100	103	145
GMTRY	74	76	78	86	116
EMIT	89	90	93	101	135
VPENTA	41	41	40	23	51
TOTAL	71	72	72	57	100

Sandia SPEED

Kernel 1	33	34	34	37	48
2	71	67	69	76	103
3	51	54	56	58	79
4	13	13	13	13	17
5	8	8	8	9	10
TOTAL	19	19	19	20	26

WHETSTONE

1 meg instr(s)	31	32	32	32	42

THE ARGONNE PROGRAMS

			X-MP SWAP	Y-MP
UNICOS4.0 CFT77(2.0)	UNICOS5.0 CFT77(3.1)	UNICOS5.1.8 CFT77 Ver 3.1.2.8	UNICOS5.1.10 CFT77 Ver 3.1.2.8	UNICO5.1.10 CFT77 Ver 3.1.2.8

LINPACK

ORD 100	28	33	29	34	46

Better LU decomposition ORD 100

UD 1	55	59	59	62	81
2	63	68	68	72	93
4	65	71	71	76	97
8	63	71	70	75	95
16	62	68	69	72	92

Better LU decomposition ORD 300

UD 1	99	104	104	108	148
2	119	125	125	133	180
4	123	129	129	141	189
8	121	131	131	142	187
16	120	128	129	140	183

UD - Unrolled depth

Livermore Loops

Kernel	UNICOS4.0 CFT77(2.0)	UNICOS5.0 CFT77(3.1)	UNICOS5.1.8 CFT77 Ver 3.1.2.8	UNICOS5.1.10 CFT77 Ver 3.1.2.8	UNICO5.1.10 CFT77 Ver 3.1.2.8
1	163	164	165	183	259
2	45	46	50	54	69
3	143	136	156	173	236
4	62	62	62	62	91
5	14	14	14	7	19
6	15	16	16	13	22
7	187	188	179	208	294
8	147	148	144	153	221
9	155	154	158	175	243
10	41	30	73	78	111
11	12	14	14	8	20
12	83	89	82	97	141
13	6	5	5	5	7
14	14	19	19	17	27
15	6	5	5	5	7
16	7	7	7	6	8
17	11	12	12	12	16
18	128	129	131	144	204
19	15	15	15	15	20
20	13	13	13	14	18
21	62	61	67	71	90
22	68	70	68	76	102
23	13	14	14	14	20
24	3	3	3	3	4

DISCUSSION OF CONFIGURATION CHANGES

(April 1989 to Present)

CFT77 (2.0) under UNICOS 4.0 to CFT77 (3.1) under UNICOS 5.0

For the most part, there is across the board improvement in the rates for CFT77 (3.1) and UNICOS 5.0 over the rates for CFT77 (2.0) and UNICOS 4.0. Each performance figure for the NAS kernels remained the same or slightly increased. Only kernel 2 in the Sandia SPEED program showed a slightly decreased mflop rate. The Whetstone rate remained the same. The LINPACK rate improved, as well as each rate for the better LU decomposition. All but two of the rates for the Livermore Loops remained the same or slightly increased. Kernel 3 of the Livermore Loops, which is a vectorized loop that does an inner product slightly decreased. The most significant performance figure is the rate for Kernel 10 of the Livermore Loops which is a little less than half of what it was for CFT77 (1.3) and previous CFT versions.

Since we could not go back and reconfigure the machine to the UNICOS 4.0 or 5.0 and CFT77 (2.0) or (1.3) configuration, and since compiler listings would not tell much, we had to spend some time investigating why the rate for kernel 10 of the Livermore Loops decreased about half. Kernel 10 is the following

```
      DO 10 L = 1,LP
      DO 10 i = 1,np
         AR      =      CX(5,i)
         BR      = AR - PX(5,i)
         PX(5,i) = AR
         CR      = BR - PX(6,i)
         PX(6,i) = BR
         AR      = CR - PX(7,i)
         PX(7,i) = CR
         BR      = AR - PX(8,i)
         PX(8,i) = AR
         CR      = BR - PX(9,i)
         PX(9,i) = BR
         AR      = CR - PX(10,i)
         PX(10,i)= CR
         BR      = AR - PX(11,i)
         PX(11,i)= AR
         CR      = BR - PX(12,i)
         PX(12,i)= BR
         PX(14,i)= CR - PX(13,i)
         PX(13,i)= CR
   10 CONTINUE
```

This loop has no trouble vectorizing. There are three scalar temporaries, AR, BR, and CR present and the increment limit np is 101. Arrays CX and PX are dimensioned CX(25,101) and PX(25,101) and hence there are no bank conflicts. We discussed this kernel with Jim Kohn who is in the FORTRAN Quality group at CRAY in Mendota Heights. His group has been using the 24 kernel Livermore Loops program to benchmark their compiler upgrades. According to his data they did not observe the performance degradation in this loop with any

previous UNICOS and CFT77 configurations. However, he observed the characteristic that there is a significant amount of alternate loading and storing being done simultaneously, whereas most vectorized loops are dominated by loads. Possibly, bidirectional memory, which allows loading and storing to be done simultaneously, may have been disabled. Several tests were conducted, on our present system configuration of CFT77 (3.1.2.8) under UNICOS 5.1.10 with the CRAY X-MP 28 model 426 to see if we could produce a MFLOP decrease of about 50 percent in the above kernel. First we changed the target configuration and compiled with nobdm (no bidirectional memory). However, this produced incorrect answers for the reason that CFT77 compiled without the safety features to prevent memory overlaps while the hardware still ran with bidirectional memory enabled. Next, we altered the cpu with /etc/cpu 01 bdmoff followed by the run of the object code. This produced a decrease of about 40 to 50 percent in the MFLOP rate whether we complied with or without the safety features. Even in dedicated mode the rate for kernel 10 dropped significantly without significantly altering the rates for the other kernels. Thus, this may have been the problem.

CFT77 (3.1) under UNICOS 5.0 to CFT77 (3.1.2.8) under UNICOS 5.1.8

Most of the performance figures remained stable with the change to CFT77 version 3.1.2.8 and UNICOS 5.1.8. The rates for the NAS kernels, Sandia SPEED, and Whetstone programs either remained the same or slightly increased. In the Argonne programs, the LINPACK rate went down slightly but the rates for the better LU decomposition remained the same. Note that in the Livermore Loops program, the rate for kernel 10 is back up to slightly higher than the rate under CFT77 version 1.3 and any previous CFT version. Except for the rates for kernels 7, 8, and 12, which decreased slightly, all rates for other kernels remained the same or slightly increased.

CFT77 (3.1.2.8) under UNICOS 5.1.10 on the upgraded CRAY X-MP

This was our first look at the performance figures under a configuration with different hardware, when the CRAY X-MP 24, model 128 was replaced with a CRAY X-MP 28, model 426. Differences include a faster CPU (8.5ns clock cycle as opposed to 9.5ns clock cycle), memory size twice as large, but a bank memory fetch that takes twice as long. The faster CPU had a positive effect on certain performance figures while the memory fetch time had a negative effect on some rates.

Note first the NAS kernels. There is about an 8 to 10 percent increase in kernels MXM, GMTRY, and EMIT, but almost a 40 percent decrease in CFFT2D and between a 40 and 50 percent decrease in VPENTA. The problem with CFFT2D and VPENTA, which was not noted until now is that they have a worse case bank conflict situation. The X-MP 28 has 32 banks. Kernel CFFT2D, which does a complex two-dimensional fast Fourier transform, has an array X dimensioned X(128,256). Kernel VPENTA, which simultaneously inverts three matrix pentadiagonals, has six two-dimensional arrays dimensioned (128,128) and two three-dimensional arrays dimensioned (128,128,3). Since the second index is changing the fastest in the vectorized loops and the leading dimension 128 is a multiple of 32, the number of banks, bank conflicts exist and they seriously degrade the performance of the kernels. They existed on the X-MP 24 also when the rates were 55 MFLOPS and 40 MFLOPS respectively for CFFT2D and VPENTA. However, the degradation is worsened on the X-MP 28 since the bank memory

fetch time is twice as long, thus accounting for the 40 to 50 percent drop. This is easily corrected by changing the leading dimension of the arrays from 128 to 129. We ran the NAS kernels in non-dedicated mode with this leading dimension change and got a MFLOP rate of 102 for CFFT2D and a MFLOP rate of 130 for VPENTA.

There was essentially little change in the Sandia SPEED and Whetstone programs. Kernel 2 in Sandia SPEED had the most improvement, about 8 percent.

In the Argonne programs, LINPACK shows a rate near what is was under CFT77 (1.3) and COS 1.15BF2. The LU decomposition rates have all increased roughly 5 percent or less.

Increases and decreases are evident in the Livermore Loops. Twelve of the kernels show rate increases, the most significant ones being Kernels 1, 18, and 22 which had increases of about 10 percent. The most significant decreases are in kernels 5 and 11, which dropped about 50 percent. Each of these has a worst case recursion situation. Kernel 6 also dropped down but not as much as kernels 5 and 11. It also has a recursion situation. Recursion itself can cause bank conflicts and since the new hardware has memory fetch taking twice as long than the previous hardware, this accounts for the decrease of about 50 percent in the kernels 5 and 11. We verified this with the following analysis. Consider kernel 5,

```
      DO 5 L = 1, LP
      DO 5 I = 2, N
5         X(I) = Z(I) * (Y(I) - X(I-1))
```

This loop does not vectorized. As X(I) is being stored, the loop index is incremented by 1 and the same value must then be retrieved (from the same bank) to be used on the right hand side in the next pass. We conducted a test on the following program with CFT77 (3.1.2.8) under UNICOS 5.1.10,

```
      T1 = SECOND()
      DO 5 I = 2, N
5         X(I) = Z(I) * (Y(I) - X(I-1))
      T2 = SECOND()
      TIME1 = T2 - T1
      T1 = SECOND()
CDIR$ novector
      DO 10 I = 2, N
10        X(I) = Z(I) * (Y(I) - W(I-1))
      T2 = SECOND()
      TIME2 = T2 - T1
```

A value of 1001 was used for N. The first loop is the same as kernel 5. The second loop is identical except with the second appearance of array X replaced with array W thus removing the recursion. The novector directive was inserted because the second loop vectorizes and runs about 13 times faster than the first loop. However, with the novector directive, neither of the loops vectorizes but the second one runs between 1.5 and 2.0 times as fast (no bank conflicts). We even went a bit further and tested the loop

```
      DO 15 I = 3, N
15        X(I) = Z(I) * (Y(I) - X(I-2))
```

and

```
      DO 20 I = 4, N
20        X(I) = Z(I) * (Y(I) - X(I-3))
```

to see if we could eliminate the effect of bank conflict in recursion. The DO 20 loop seemed to totally eliminate the effect as it ran almost twice as fast as the DO 5 loop. Note that with the index I-3 in the second appearance of X, the same bank will be accessed on ever third execution of the loop instead of every other execution with index I-2 or every execution with index I-1.

Y-MP with CFT77 version 3.1.2.8 under UNICOS 5.1.10

For comparison purposes, we thought it interesting to show the performance rates for the Y-MP model 8/6128 (6 processors and 128 M words of memory). As expected, there is across the board improvement in the rates and particularly significant improvement in those kernels which vectorize well. The Y-MP has a 6 ns CPU clock as opposed to an 8.5 ns CPU clock on the X-MP. In some kernels there is not much difference but that is because they are not able to demonstrate what this kind of architecture can do. For example, kernel 16 in the Livermore Loops is a Monte Carlo Search loop which has a lot of arithmetic IF(s), and kernel 24 is a small loop that searches for the minimum in an array and is dominated by logical testing.

DISCUSSION OF THE LINPACK RATES

Some remarks concerning the LINPACK rates should be made, especially in relation to the figures reported in the report "Performance of Various Computers Using Standard Linear Equations Software" by Jack Dongarra dated October 1, 1990. In that report, the MFLOPS figure for LINPACK, n = 100, on a CRAY X-MP/416 (one processor, 8.5 ns clock) is 70. Furthermore, the figure for a CRAY X-MP/416 (two processors, 8.5 ns clock) is 115. For a Y-MP/832 (6 ns clock) with 1, 2, and 4 processors the rates are respectively 90, 144, and 226.

The figures shown in this report are not nearly that high. According to ground rules for running the benchmark, no changes are to be made to the FORTRAN source, not even changes in the comments. Therefore, we made no attempt to inline subroutines, or to use automatic microtasking features of the compiler.

The figures reported in [2], were obtained by inlining four of the subroutines in the program. The subroutines are SAXPY, SDOT, SSCAL, and ISAMAX, which are in turn called by SGEFA and SGESL. Also, the source was compiled under CF77 (version 4.0) with the option -Zp, which causes autotasking and microtasking to be done. In case of more than 1 CPU, wall clock time was used instead of CPU time to calculate MFLOPS.

By inlining the four subroutines identified above and using the -Zp option for more than one CPU, we were able to obtain the following rates. For 1 and 2 CPUs on the X-MP, 67 MFLOPS and 105 MFLOPS were obtained respectively.

For 1, 2, and 4 processors on the Y-MP, 88, 141 and 218 MFLOPS were obtained respectively.

FINAL DISPLAY OF FIGURES

At the end of this writing, our X-MP and Y-MP system configurations were just upgraded to UNICOS 6.0 and CFT77 version 4.0.3. The figures that follow show the rates for that figuration, obtained early July, 1991.

	X-MP	Y-MP
Program	UNICOS 6.0 CFT77(4.0.3)	UNICOS 6.0 CFT77(4.0.3)

NAS Kernels

MXM	193	272
CFFT2D	36	73
CHOLSKY	63	88
BTRIX	102	142
GMTRY	84	112
EMIT	129	175
VPENTA	25	55
TOTAL	58	103

Sandia SPEED

kernel 1	36	48
2	76	102
3	57	78
4	60	83
5	9	11
TOTAL	27	35

WHETSTONE

1 meg instr(s)	30	40

THE ARGONNE PROGRAMS

X-MP	Y-MP
UNICOS 6.0 CFT77(4.0.3)	UNICOS 6.0 CFT77(4.0.3)

LINPACK

ORD 100	34	37

Better LU decomposition ORD 100

UD 1	62	79
2	71	91
4	76	95
8	73	92
16	69	88

Better LU decomposition ORD 300

UD 1	108	145
2	132	178
4	140	184
8	139	182
16	133	175

UD - Unrolled depth

Livermore Loops

	X-MP	Y-MP
Kernel	UNICOS 6.0 CFT77(4.0.3)	UNICOS 6.0 CFT77(4.0.3)
1	181	258
2	53	65
3	173	236
4	60	90
5	13	19
6	13	21
7	207	295
8	161	230
9	175	242
10	77	109
11	13	20
12	97	142
13	5	7
14	17	27
15	21	30
16	6	8
17	12	16
18	144	203
19	14	20
20	14	18
21	68	84
22	76	100
23	14	20
24	3	4

DISCUSSION OF THE FINAL FIGURES

According to CRAY release notes, many problems that existed with CFT77 were fixed with CFT77 version 4. Also, enhancements included the ability to vectorize more types of loops. This is reflected in the rates seen for kernel 4 in the Sandia SPEED program and in loop 15 in the Livermore Loops which showed the most significant changes. Kernel 4 in the Sandia SPEED program went from 13 to 60 on the X-MP and from 17 to 83 on the Y-MP. It has an inner loop with conditional branching, which did not vectorize under previous versions of CFT77 but does vectorize under CFT77 (4.0.3). Loop 15 in the Livermore loops also has an inner loop with conditional branching which now vectorizes under CFT77 (4.0.3). Its rates increased by a factor of four.

In the NAS kernels, the rate for MXM went down slightly, but the rate for EMIT went up.

Aside from the significant increase in the rate for kernel 4 explained above, the rates for the Sandia SPEED kernels essentially remained the same.

Whetstone rates did not improve.

For the Argonne programs, the LINPACK rate on the Y-MP went down slightly, and the rates for the larger unrolled depths in the better LU decomposition went down slightly. We did not investigate specific reasons for this.

The vector loop program did not produce any change from the results obtained with CFT77 version 2.0.

In the Livermore loops, note the rates for kernels 5 and 11, which have worse case recursions and had an effect on the X-MP with the hardware upgrade. These rates have improved to what they were before the hardware change. Thus the compiler has improved its ability to handle certain types of recursion.

CONCLUSION

Overall, the figures show that MFLOP rates have significantly increased through system configuration upgrades. Rates can remain unchanged and they can go down as well as up. Although reasons for most changes in the rates can be explained by compiler changes or upgrades, performance changes can also be caused by system and hardware changes.

REFERENCES

1. Huss, J.E.; and Pennline, J.A.: A Comparison of Five Benchmarks. NASA TM-88956, 1987.

2. Dongarra, J.J.: Performance of Various Computers Using Standard Linear Equations Software. Computer Science Department, University of Tennessee, Knoxville, TN, October 1, 1990.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (*Leave blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1991	Technical Memorandum

4. TITLE AND SUBTITLE
Cray Performance Data From Five Benchmarks

5. FUNDING NUMBERS
WU-NONE

6. AUTHOR(S)
James A. Pennline

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135-3191

8. PERFORMING ORGANIZATION REPORT NUMBER
E-6503

9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-105200

11. SUPPLEMENTARY NOTES
Responsible person, James A. Pennline, (216) 433-5058.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

MFLOPS performance data for five benchmark programs, discussed in TM-88956, is reported for different system configurations on a CRAY X-MP. The differences in the system configurations are results of compiler and/or system upgrades, if not a hardware upgrade. The effects of the changes in the system configuration on the performance figures are analysed and discussed.

14. SUBJECT TERMS
Cray computers; Compilers; Performance; Operating systems

15. NUMBER OF PAGES
24

16. PRICE CODE
A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	