SANDIA REPORT SAND81-2494 • Unlimited Release • UC-32
Printed April 1982

Methods of Reducing Program Execution Time Under RT-11 FORTRAN

Robert J. Isidoro, Ronald E. Trellue

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550
for the United States Department of Energy
under Contract DE-AC04-76DP00789

SF2900Q(8-81)

DISCLAIMER

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Printed in the United States of America
Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

NTIS price codes
Printed copy: A02
Microfiche copy: A01

SAND81-2494
Unlimited Release
Printed April 1982

Distribution
Category UC-32

Methods of Reducing Program-Execution Time Under RT-11 FORTRAN

Robert J. Isidoro
Flight and Lab Test Development Division III, 1423

Ronald E. Trellue
Minicomputer Software Division 1523
Sandia National Laboratories
Albuquerque, New Mexico

SAND--81-2494

DE82 015658

Abstract

A requirement to double the number of data channels for our RT-11 data acquisition and processing system on a PDP 11/34 with RK05 disks from 64 to 128 channels led us to consider reducing the size of the buffers used to sort the data since the program size had already reached the upper limit. But the reduction in buffer size could result in a significant increase in processing time. It was decided to conduct a benchmark study to determine if processing time could be improved by using virtual arrays to optimize buffer size. Additional test runs were performed utilizing cache memory, a floating point processor, different disks, and various processors. The results of this study should provide our management with sufficient data to determine the cost-effectiveness of buying additional memory, or other improvements to the present RT-11 systems.

Contents

Figures

Tables

Methods of Reducing Program Execution Time Under RT-11 FORTRAN

Introduction

The Quality Assurance (QA) Department of Sandia National Laboratories is responsible to the Department of Energy for assurance that weapons remain functional throughout their stockpile life. To accomplish this, QA conducts laboratory system tests on the Sandia-designed components of the weapon system. Joint flight tests with the Department of Defense are also conducted. The data acquisition and processing system used to acquire and analyze test results was designed by the QA Systems Test Equipment Design Division. The acquisition systems are built around PDP 11/34 computers. There are six similar acquisition systems that collect data independently from many unique weapon testers. (See "STATS—A Unique High Speed, Multiple Channel, Real-Time Data Acquisition System" in the *Proceedings of the Digital Equipment Computer Users Society*, Vol. 7, No. 2, Fall 1980 for details.) Data is collected and compressed in real-time at high speeds and then written to disk. The RT-11 Single-Job Monitor is used along with an assembly language acquisition program to collect the data. A test usually lasts several minutes. After the data is acquired, the system engineers are interested in seeing the results as soon as possible.

The complete test analysis must be known before disassembling the test equipment and moving on to the next scheduled test. If anomalies were present, disassembling would compromise posttest troubleshooting procedures. The analysis for each test is therefore performed on the acquisition machine immediately after each test and must be completed in as short a time as possible.

The FORTRAN software package used to analyze the results of laboratory system tests is the subject of this paper. We will discuss in general how the software works, problems encountered when we decided to double the number of data acquisition channels to analyze, and the solution to the problems arrived at by benchmarking the programs with optional equipment that could be added to the existing configuration.

Configuration

Our acquisition systems are built around the following hardware and software:

PDP 11/34 with 32K words of memory
Dual RK05 disks, one for the system, one for data
RX01 floppies for temporary storage
CRT terminal at 9600 baud
Versatec printer/plotter
RT-11 single-job monitor
FORTRAN IV application programs
FORTRAN inline code—Extended Instruction Set (EIS)

What the Program Does

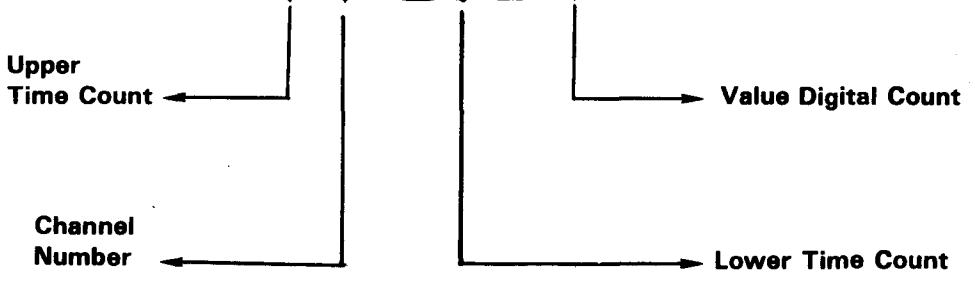
In order to see what the data analysis program does, let us look first at the input data collected during the test. Then we will look at the output from the analysis program to see what has been accomplished. The data is compressed in real time by the acquisition system and stored on a removable RK05 disk.

Figure 1 shows a typical block of data created by the acquisition program. Each data point within the block occupies three words (16 bits each) and consists of information concerning

1. Identification of the channel number associated with the data
2. A digital count to be converted to a time
3. A digital count to be converted to a value in engineering units (i.e., volts, amps, etc)

Each block contains data points from many different channels. This will make it necessary to sort the file by channel before analysis can be performed. Looking at this file, one can see that most engineers would be unhappy with this as the final output. Engineers want to see the times expressed in seconds, values expressed in units they understand, and some additional analysis to assure them that test requirements for the unit under test were actually met.

```
DY:WESD3R.104/0:0
BLOCK NUMBER  000000
000/ 000007 000000 013776 000010 000000 013776 000012 000000 *....^.....^.....*
020/ 014000 000013 000000 014000 000015 000000 014013 000016 *................*
040/ 000000 014010 000020 000000 013777 000022 000000 016355 *..............m.*
060/ 000023 000000 013777 000024 000000 014000 000025 000000 *................*
100/ 016171 000007 014206 033757 000010 014206 033757 000012 *y.....o7....o7..*
120/ 014207 073764 000007 014210 133730 000010 014210 133732 *..tw....X7....Z7*
140/ 000012 014213 033776 000007 014211 053711 000010 014211 *....^7....IW....*
160/ 053714 000013 014223 173757 000016 014225 043767 000020 *LW....ow....wG..*
200/ 014224 114024 000022 014224 116321 000023 014225 074032 *........Q......x*
220/ 000024 014225 034117 000007 014213 153731 000010 014213 *....O8....YW....*
240/ 153727 000012 014225 153762 000013 014224 114040 000016 *WW....rW.... ....*
260/ 014226 104006 000020 014225 034713 000022 014225 035677 *........K9....?;*
300/ 000023 014226 014616 000024 014225 155177 000007 014214 *............Z....*
320/ 073757 000010 014214 073756 000012 014227 014217 000013 *ow....nw........*
340/ 014225 035100 000020 014225 156466 000022 014225 155226 *..@:....6]....Z*
360/ 000023 014226 136237 000024 014226 076677 000007 014215 *.....<....?}....*
400/ 013773 000010 014215 013772 000012 014227 135453 000013 *{.....z.....+;..*
420/ 014225 156540 000020 014226 077424 000022 014226 075643 *..`]..........#{*
440/ 000023 014227 057034 000024 014227 017537 000007 014225 *.....^....._.....*
460/ 034067 000010 014224 114006 000012 014230 056657 000013 *78........../]..*
500/ 014226 077361 000020 014227 017376 000022 014227 016661 *..q^....^.....1.*
520/ 000023 014227 176430 000024 014227 137602 000007 014225 *.....}.....?....*
540/ 154777 000010 014225 034054 000012 014230 176702 000013 *.Y....,8....B}..*
560/ 014227 017464 000020 014227 137340 000022 014227 137376 *..4.....`>....^>*
600/ 000023 014230 116337 000024 014230 057617 000007 014226 *....._.....^....*
620/ 076377 000010 014225 154742 000012 014231 116247 000013 *.I....bY....'...*
640/ 014227 137533 000020 014230 057515 000022 014230 057561 *..[?....M_....q_*
660/ 000023 014231 036677 000024 014230 177661 000007 014227 *....?=....1.....*
700/ 017477 000010 014226 076407 000012 014232 036306 000013 *?.....}....F<..*
720/ 014230 057607 000020 014230 177625 000022 014230 177616 *...._.............*
740/ 000023 014231 157350 000024 014232 037634 000007 014227 *....h^.....?....*
760/ 137715 000010 014227 017416 000012 014232 157035 000013 *M?..........^..*
```



NOTES:

1. **Each data point consists of three 16 bit words.**

2. **Each block of data contains data points from many different data channels.**

Figure 1. Sample Block From Raw Data File

Let us now look at Figure 2 which is an example of a typical output to the line printer from the data analysis program. The header information at the top contains the descriptive information necessary to identify the test being performed. This is followed by a statement that a key event during the test (event 112) was missed. Then we see a set of answers to questions concerning how well the unit under test performed. These questions are contained in an analysis file created by the engineer. This file contains all those requirements he considers necessary to assure himself that the unit under test either did or did not perform satisfactorily.

Each output line contains

- A numerical identifier
- The value obtained from the analysis
- A description of the value
- Upper and lower limits for satisfactory performance
- A diagnostic H or L indicating the value obtained from the analysis was either "High" or "Low" when compared to the limits.

Although there are none displayed, diagnostic statements can be displayed (such as SWITCH DID NOT CLOSE) when requirements are not met. At the bottom of Figure 2 is a listing of the times that key events during the test were observed. Each time is associated with its corresponding event number.

How the Program Works—The Mix

The program is quite large. In fact, it became apparent during the development phase that we would have to take advantage of things like overlays and chaining to make it fit into the available 24K words of memory provided by RT-11 for FORTRAN programs.

As shown in Figure 3, the core program (DATZ) initializes things, opens up the necessary files, and calls in PASS1 as an overlay. PASS1 then reads through the raw data file on RK05 and counts the number of data points on each channel present. It then computes the starting record number for each channel in the direct-access file to be created in PASS2 based upon the total number of readings for each channel. Most of the execution time in PASS1 is probably spent in disk-to-memory accesses.

```
                                WEAPON TEST REPORT

                        TIME:14:40:36                DATE:12-NOV-81

                                HEADER INFORMATION

 M.I.   CYCLE  MATL  SERIAL  TEST   MFGR.  ATT  SOURCE  ALT    TEST     PART   PN     SEQ   CONTROL  MISCELLANEOUS  BA DWG.  ISS
NUMBER   NO.   TYPE  NUMBER  I.D.   CODE   NO.  CODE    CODE   DATE     NUMBER SUFFIX  NO.   FILE ISS.              NUMBER

301459    01    SPE     1004  WESD3  Q      1    Q              07/19/79 301459 00      04      A                      301459    H

                    ********** EVENT 112 IS MISSING **********  <-- From EPASS

400.  9.05515  SECONDS   HV    APPLY TIME                    9.0     9.1
401. 0.148000 SECONDS   HV MONITOR DIV RISE TIME            0.1     0.5
402. 0.146800 SECONDS   HV MONITOR AMP RISE TIME            0.1     0.5
403.  16.4685 VOLTS      HV MONITOR DIVIDER AVG             12.9    17.3
404.  7.70996 VOLTS     HV MONITOR AMP AVG                  7.0     9.0
407. 0.221801  SECONDS  POLING VOLTAGE RISE TIME            0.1     0.5
408.  462.958  VOLTS    POLING VOLTAGE                      450.0   550.0
410.  30.1456  MILLIAMPS POLING CURRENT CHARGE MAX          20.0    40.0
411. 0.113818  MILLIAMPS POLING CURRENT STEADY AVG          0.0     1.0
430.  5.05157  SECONDS  HV TO A3 TIME                       5.0     5.1
432.  25.1558  VOLTS    A3 IN MAX                           4.0     16.0    H
433. -0.52406  MILLIAMPS A3    PEAK CURRENT                 40.0    56.0    L
435.  1.34277  VOLTS     A4 TC MAX                          -10.0   10.0
436. 0.854492  VOLTS     A4 IN MAX                          -10.0   10.0
437. 0.000000  VOLTS     A4   E-GEN POWER MAX               -10.0   10.0
438.  25.4092  VOLTS     A1/A3 PS AVG                       25.0    26.0
439.  467.696  VOLTS     POLING PS AVG                      450.0   550.0

                                                  <-- Answers to Analysis File Questions From DTFQD

                                EVENT TABLE

NUMBER      TIME          NUMBER      TIME          NUMBER      TIME          NUMBER      TIME          NUMBER      TIME

  71        9.0552          72        9.2031          81        9.0582          82        9.2050         101       14.1071
 102       14.6106         111       14.1067         113       14.1088         191        9.0553        192        9.2772
 201        9.0525         202        9.2955

                                <-- Event Table From EPASS
```

Figure 2. Weapon Test Report

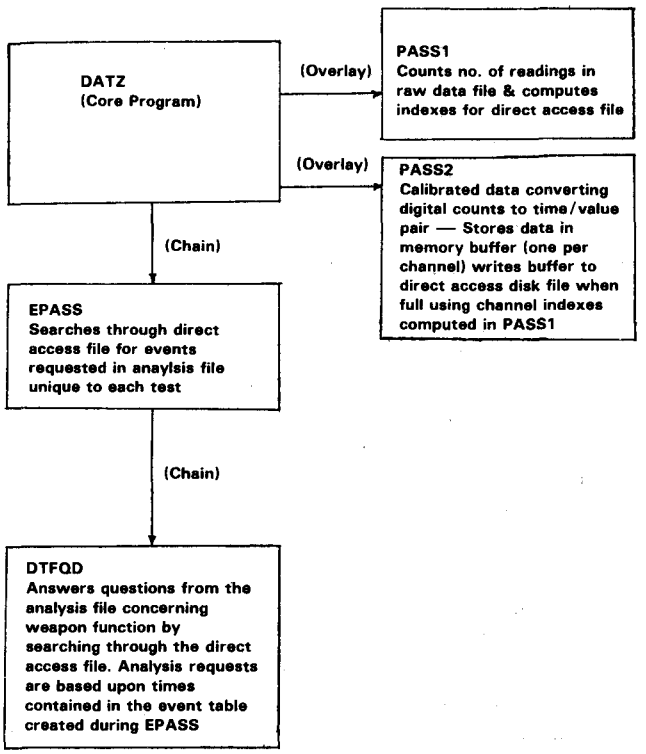


Figure 3. Program Mix

PASS2 is then called in as an overlay. PASS2 calibrates the digital data converting it to time/value pairs and then stores the data in memory buffers, one for each channel. These memory buffers must be large enough to make the program run in a reasonable amount of time when large data files are encountered, and yet small enough to reside in the available amount of memory. When the data buffer for a given channel becomes full, PASS2 writes the data into a direct access file sorted by channel number on the RK05 using channel indexes computed in PASS1 to determine the correct record number. There are memory-to-disk accesses associated with writing the filled memory buffers to the direct-access file on the RK05. PASS2 execution time depends primarily on the number of disk access in both reading from the raw data file and writing to the sorted data file.

When all the data has finally been written onto the sorted data file in PASS2, DATZ chains to EPASS. EPASS then searches through the newly created file looking for events requested by the engineer in his analysis file. In EPASS the execution time depends upon the number of reads from disk required to establish each event and the number of events requested in the analysis file.

When the event search has been completed, EPASS will chain to the DTFQD program which answers questions from the analysis file concerning weapon function by searching through the direct-access file. Analysis requests are based upon times contained in the event table created during EPASS. In DTFQD some calculations are required to determine average values, or time/value products within a prescribed time interval, but the execution time depends mainly upon the number of disk accesses required to answer each question asked and the total number of such questions in the data analysis file.

Problem

The problem we encountered occurred when system requirements dictated that we be able to process data from up to 128 channels. Previously we had only been able to process data from up to 64 channels at a reasonable rate. The main problem with going to 128 channels was one of program size. The program was pressing memory limits already even though it had been broken up into separate programs that were chained together, and the individual programs were overlayed. The parameter which had to give in order to handle 128 channels was the buffer size, or the amount of data to read in from the disk at one time. As can be seen from Figure 4, the program must read in the data from the raw data file, calibrate and convert times and values to floating point numbers, store the result in the appropriate memory buffer (one per channel), and finally when a memory buffer becomes full, write the information into the appropriate record of the sorted data file. It was found that with the memory available (24K words), one could use a buffer size of 64 floating point numbers (128 words) per channel as long as there were only 64 channels. To go to 128 channels would require reducing the buffer size to 32 words. This caused concern that processing time would be significantly increased.

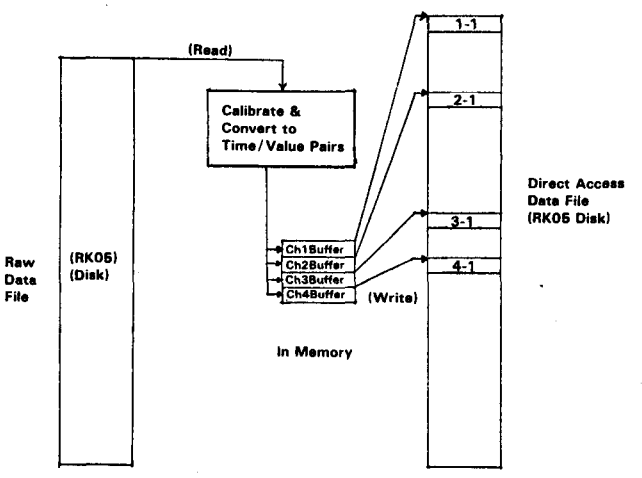


Figure 4. PASS2 Diagram

Goal

Our goal was to find an economical way to process raw data files with up to 128 channels, and to reduce the total analysis time as much as possible for each real test. We wanted to find an economical solution since it would have to be implemented on six similar acquisition machines. The first solution tried was virtual arrays. In addition to virtual array tests, other timing tests were run to evaluate the effects of different disk types, cache memory, the addition of a floating point processor, and different types of CPUs.

Datasets

Timing tests were run with three different sets of raw data files representing a short, medium, and long analysis process. These files are identified by the names WESD3, JNOD3, and WHIGE respectively. The raw data files used in these tests are characterized by the following parameters:

	WESD3	JNOD3	WHIGE
Number of blocks	22	114	256
Number of event requests	13	18	311
Number of analysis requests	17	20	251

As can be seen from the parameters, the "short" test has fewer blocks in the raw data file, seeks fewer events, and asks fewer questions concerning test performance than the "medium" and the "long" tests. Processing time for a given test is a function of the number of disk accesses required for each event or analysis request, and the total number of such requests made.

Unless otherwise stated, each timing test was run on a machine with similar hardware and software. All line printer output was diverted to the terminal at 9600 baud (with JUMP MODE for VT100) to eliminate the variation due to different printer speeds.

Situation

The times shown in Table 1 illustrate the processing times of the three test sets on the PDP 11/34 with 128 word buffers but only supporting 64 channels. This is the original configuration which was working well until more channels needed to be supported. In all of the tables the time given for the test runs is in minutes:seconds, and the Buffer = parameter is the number of words allocated. The time indicated by the number (2) in Table 1 is the time required to support 128 channels, which would mean dropping the buffer size to 32 words. For the three tests, this resulted in an increase of about 20% in the processing time. With JNOD3, for example, changing the buffer size from 128 words to 32 words in order to service 128 channels, the analysis time would increase from 2 minutes, 35 seconds to 3 minutes, 14 seconds.

Table 1. PDP 11/34 Original Configuration

(1) Buffer = 128, Channels = 64 vs
(2) Buffer = 32, Channels = 128

File	Time	Percent
WESD3	(1) 0:56	
	(2) 1:07	20
JNOD3	(1) 2:35	
	(2) 3:14	25
WHIGE	(1) 7:52	
	(2) 9:13	17

Virtual Arrays

One solution to our problem was thought to be virtual arrays. This could be a possible solution since all of the programs running the analysis were written in FORTRAN under RT-11. There were several questions concerning virtual arrays, such as would they work, what overhead would be incurred, would they speed up processing an acceptable amount, and what was the cost.

PDP 11/34 vs PDP 11/45

A number of timing tests were run on a PDP 11/45 with 128K words of core memory in order to try various virtual array schemes. We needed to use the PDP 11/45 because at the time our PDP 11/34s had only 32K words of memory. Table 2 indicates that the PDP 11/45 ran 10% to 12% faster than the PDP 11/34. Later this information allowed us to correlate the final results back to projected run times on the PDP 11/34.

Table 2. PDP 11/34 vs PDP 11/45 (No Virtual)

(1) PDP 11/34 Buffer = 128 (64 channels)
(2) PDP 11/45 Buffer = 128 (64 channels)
(3) PDP 11/34 Buffer = 32 (128 channels)
(4) PDP 11/45 Buffer = 32 (128 channels)

File	Time	Percent
WESD3	(1) 0:56	
	(2) 0:56	0
	(3) 1:07	
	(4) 1:00	10
JNOD3	(1) 2:35	
	(2) 2:17	12
	(3) 3:14	
	(4) 2:53	11
WHIGE	(1) 7:52	
	(2) 6:54	12
	(3) 9:13	
	(4) 8:26	8

Virtual Array Overhead

How much overhead would be introduced going to virtual arrays? Table 3 shows the overhead in going from regular arrays to virtual. These test were all run on the PDP 11/45.

Table 3. Virtual Array Overhead

(1) No virtual, Buffer = 128
(2) Virtual, Buffer = 128
(3) No virtual, Buffer = 32
(4) Virtual, Buffer = 32

File	Time	Percent
WESD3	(1) 0:56	
	(2) 0:57	2
	(3) 1:00	
	(4) 1:07	2
JNOD3	(1) 2:17	
	(2) 2:23	4
	(3) 2:53	
	(4) 3:04	6
WHIGE	(1) 6:54	
	(2) 7:04	2
	(3) 8:26	
	(4) 8:35	2

The overhead in going to virtual arrays for our application was 2% to 4%. We had heard that virtual array overhead was on the order of 10%. Since our application entailed more than just array accesses, the overall effect of changing a regular array to a virtual array was minimal.

Optimizing Buffer Size

Virtual arrays provided two benefits right away to our programs. First, the program was able to process 128 channels of data. Secondly, increasing the size of our buffers would proportionally decrease the number of disk I/Os, therefore speeding up the analysis for all live tests. The question of how large a buffer to use would dictate the amount of memory necessary to buy. Table 4 illustrates the various buffer sizes tested and the results obtained.

Table 4. Buffer Optimization

PDP 11/45 virtual arrays, 128 channels
(1) Buffer = 32, (2) Buffer = 128
(3) Buffer = 256, (4) Buffer = 508

File	Time	Percent	No. of Disk I/Os
WESD3	(1) 1:07	—	262
	(2) 0:57	15	88
	(3) 0:57	15	49
	(4) 0:54	19	37
JNOD3	(1) 3:04	—	243
	(2) 2:23	22	152
	(3) 2:15	27	130
	(4) 2:24	22	111
WHIGE	(1) 8:35	—	6741
	(2) 7:04	18	1643
	(3) 6:42	22	844
	(4) 6:54	20	441

Increasing the buffer size from 32 to 128 words resulted in a decrease in time of about 18%, and increasing buffers from 32 to 256 words seemed to improve the processing speed by about 21%. These times apply to running on the PDP 11/45 which is 10% faster than the PDP 11/34. All times shown in Table 4 would have been greater on the target machine, the PDP 11/34, but the percentage of change would probably remain the same. The main reason for faster execution with larger buffers is shown in the far right column of Table 4 by the number of disk I/Os

effected by the changes in buffer size. The speed improvement in going to 256 word buffers over 128 was not phenomenal, but would be easy to implement with relatively low cost since additional memory would be required to support the 128 word buffers anyway. The reason that a buffer size of 508 words actually caused a decrease in processing time is harder to explain. The answer is based in the type of data to be analyzed, and the type of analysis being done. A large amount of processing time is spent searching through the data for particular events. In reading in large amounts of data, more information than necessary could be read in before the event was found.

Virtual Array Test Results

Virtual arrays seem to provide a solution to the problem we faced. First of all, virtual arrays did indeed work, and were easy to implement. The only change required to the program was to replace the DIMENSION statement with a VIRTUAL statement. Fortunately no changes were needed to accommodate any of the virtual array restrictions.

By going to virtual, overall program size was expanded to handle up to 128 channels without seriously effecting processing time. In fact, by increasing the buffer size to 256 words, we project processing times to actually decrease by about 20% over the time which would be required to process the same data with 32 word buffers, without the additional memory. Testing showed that an optimum buffer size could be found, and that bigger does not necessarily mean better. The price for 32K words of MOS memory ran about $1100 per machine for a total cost of about $6600.

Other Options

RK05 Disk vs RL01 Disk

One alternative which was investigated was to upgrade from RK05 to RL01 disks. This move would not help decrease program size, but might speed up run times. Table 5 itemizes the results. All runs were on the PDP 11/45 which has both RK05 and RL01 disks.

RL01 disks are not only larger than RK05s, but also faster. On the shorter tests which do a minimal amount of processing, RL01s speeded things up about 19%. As the buffer size increases, the number of disk I/Os drops dramatically and the effect of the faster disks diminishes to about 13% for JNOD3, and to 6% for WHIGE. The cost to upgrade the six PDP 11/34 acquisition systems to dual RL01s would have been about $55,000. It was pleasantly surprising to learn that the RL01 showed an appreciable decrease in processing time; this fact will be considered if a disk upgrade is required at some later date.

Table 5. RK05 vs RL01

Buffer = 32 (1) RK05, (2) RL01
Buffer = 128 (3) RK05, (4) RL01
Buffer = 256 (5) RK05, (6) RL01

File	Time	Percent
WESD3	(1) 1:00	
	(2) 0:50	17
	(3) 0:56	
	(4) 0:45	20
	(5) 0:57	
	(6) 0:46	19
JNOD3	(1) 2:53	
	(2) 2:25	16
	(3) 2:17	
	(4) 2:01	12
	(5) 2:15	
	(6) 1:58	13
WHIGE	(1) 8:26	
	(2) 7:24	12
	(3) 6:54	
	(4) 6:31	5
	(5) 6:42	
	(6) 6:16	6

RL01 vs Winchester Disk

Another possible option that was examined was going to Winchester disks. This type of disk provides large amounts of data storage at a low price. Tests were run on a LSI 11/23 with RL01 disks, and then a similar LSI 11/23 with a DSD 880 Winchester disk. This disk emulates a single RL01. The results are summarized in Table 6.

These results show that the Winchester disk was a few percentage points slower than the RL01. Compared with Table 5, Table 6 indicates that the Winchester is faster than an RK05. On the longer runs which require a smaller percentage of the time for I/O, the Winchester is almost as fast as the RL01. The price of a Winchester is about $6500 for 10MB. Six of them would be about $39,000. We could get a 30MB or more Winchester disk for about the price of dual RL01s.

Table 6. RL01 vs Winchester

Buffer = 32 (1) RL01, (2) Winchester
Buffer = 128 (3) RL01, (4) Winchester
Buffer = 256 (5) RL01, (6) Winchester

File	Time	Percent
WESD3	(1) 0:58	
	(2) 1:04	9
	(3) 0:52	
	(4) 0:56	7
	(5) 0:54	
	(6) 0:58	7
JNOD3	(1) 2:56	
	(2) 3:07	6
	(3) 2:30	
	(4) 2:36	4
	(5) 2:32	
	(6) 2:38	4
WHIGE	(1) 8:58	
	(2) 9:15	3
	(3) 7:53	
	(4) 8:00	1.5
	(5) 7:45	
	(6) 7:51	1.3

Cache Memory

The rumors concerning the speed of cache memory prompted an investigation not so much as a solution to the processing of 128 channels, but as a method of decreasing processing time. A PDP 11/34 with cache memory was found, and the same tests were run on it. All other configuration parameters remained the same. The results are illustrated in Table 7.

Table 7 shows that the addition of cache memory to the PDP 11/34 resulted in a reduction of 20% in the processing time for the medium and long runs over the times on the original PDP 11/34. If the buffer size is small, as indicated by runs (3) and (4), the number disk I/O's is increased and the effect of the cache memory is not quite as great. These results indicate cache memory might be a viable way to solve the processing speed problem. The cost of implementing cache memory on six PDP 11/34s would be about $22,500.

Table 7. Cache Memory

No virtual, Buffer = 128, Channels = 64
(1) PDP 11/34 cache, (2) PDP 11/34 no cache
No virtual, Buffer = 32, Channels = 64
(3) PDP 11/34 cache, (4) PDP 11/34 no cache

File	Time	Percent
WESD3	(1) 0:46	
	(2) 0:56	18
	(3) 0:56	
	(4) 1:07	16
JNOD3	(1) 2:02	
	(2) 2:35	22
	(3) 2:41	
	(4) 3:14	17
WHIGE	(1) 6:04	
	(2) 7:52	22
	(3) 7:24	
	(4) 9:13	20

Floating Point Processor

Table 8 contains the results of running our analysis programs on a machine with a floating point processor using threaded code and a library built for FPU. The analysis program deals with single-precision floating point values for many of its calculations.

Table 8. Floating Point Processor

Buffer = 32 (1) PDP 11/34 cache, no FPU
(2) PDP 11/34 cache, with FPU
Buffer = 128 (3) PDP 11/34 cache, no FPU
(4) PDP 11/34 cache, with FPU

File	Time	Percent
WESD3	(1) 0:56	
	(2) 0:56	0
	(3) 0:46	
	(4) 0:47	-2
JNOD3	(1) 2:41	
	(2) 2:39	1
	(3) 2:01	
	(4) 1:59	2
WHIGE	(1) 7:24	
	(2) 7:36	-3
	(3) 6:04	
	(4) 6:08	-1

These results indicate that for our application the addition of a floating point processor would have little effect on processing time.

Different CPU Types

Table 2 has shown that the PDP 11/45 can do the same analysis in about 10% less processing time than the PDP 11/34. We ran our datasets and analysis programs on various CPU types just to see how they would perform with our application. The results are represented in Table 9. The first row (1) is the run time on the original PDP 11/34 configuration. The percentages indicate how much faster or slower the analysis ran on the other CPUs.

Table 9. Various Processors

	Buffer = 32		Buffer = 128		Buffer = 256	
WESD3						
(1) 11/34	1:07	—	0:56	—	1:03*	—
(2) 11/23	1:10	-4%	1:04	-14%	1:07	-6%
(3) 11/45	1:00	10%	0:56	0%	0:57	9%
(4) 11/34c**	0:56	16%	0:46	18%	0:46	27%
(5) 11/44	0:49	27%	0:39	30%	0:38	40%
(6) 11/60	0:54	19%	—	—	0:47	25%
JNOD3						
(1) 11/34	3:14	—	2:35	—	2:28*	—
(2) 11/23	3:29	-8%	2:50	-10%	2:53	-17%
(3) 11/45	2:53	11%	2:17	10%	2:15	9%
(4) 11/34c**	2:41	17%	2:01	22%	1:54	23%
(5) 11/44	2:15	30%	1:36	38%	1:27	41%
(6) 11/60	2:43	16%	—	—	2:03	17%
WHIGE						
(1) 11/34	9:13	—	7:52	—	7:22*	—
(2) 11/23	10:13	-11%	9:09	-16%	8:18	-13%
(3) 11/45	8:26	8%	6:54	12%	6:42	9%
(4) 11/34c**	7:24	20%	6:04	23%	5:41	23%
(5) 11/44	6:26	30%	4:58	37%	4:32	38%
(6) 11/60	7:31	18%	—	—	5:59	19%

*Projected
**Cache

The LSI 11/23 ran about 11% slower than the PDP 11/34, substantiating Digital claims that the LSI 11/23 is about 90% of a PDP 11/34. The PDP 11/45 ran about 10% faster than the standard PDP 11/34. The PDP 11/34 with cache memory processed our tests about 20% faster than the standard PDP 11/34. The PDP 11/44 executed our test runs 30% to 40% faster than the PDP 11/34. Strangely enough, the PDP 11/60 did not seem to perform much better than the PDP 11/34 with cache for our particular analysis environment. The PDP 11/60 was configured slightly different than the other hardware configurations, and the RK05 disks used for the system disk, data disk, and scratch areas were the last devices on the bus. For these reasons, PDP 11/60 times are probably not directly comparable with the other configurations. These results indicate we should give serious thought to adding cache to the PDP 11/34 or upgrading to the PDP 11/44.

Conclusions

After looking over all test results, the following conclusions were reached:

- Virtual arrays do work and are easy to implement.
- Virtual array overhead was minimal, about 3%.
- Buffer size can have an effect on processing, and can be optimized for a particular application.
- Buying additional memory for virtual array support is an economical way to extend the size of a program for about $1100 per machine.
- RL01 disks are about 13% faster than RK05s, but are an expensive upgrade at about $9000 for dual RL01s per CPU.
- A Winchester type disk, the DSD 880, is slower than an RL01, about 4%. The cost for a 10MB disk would be about $6500.
- Cache memory alone improved processing time 20% but did not allow for extending the size of the programs themselves. The cost would be about $3700 per machine.
- A floating point processor would not improve our processing times.
- The LSI 11/23 runs about 10% slower than the PDP 11/34.
- The PDP 11/45 runs about 10% faster than the PDP 11/34.
- The PDP 11/34 with cache was 20% faster than the standard PDP 11/34.
- The PDP 11/44 runs 35% faster than the PDP 11/34.
- The PDP 11/60 performance was about the same as the PDP 11/34 with cache for our application.

It was decided to go ahead and buy the additional memory for our machines and implement virtual arrays, since virtual arrays seemed to work, allowed us to go to 128 channel support, actually decreased processing time due to larger buffer sizes, and was the most economical solution. The additional memory would not only help reduce analysis time, but could be used in other stages of the testing such as calibration, data acquisition, and the plotting of the data on the Versatec printer/plotters.

The virtual array scheme discussed in this paper was implemented shortly before the final draft was typed, and we were able to compare our projected run times with the actual result. For WESD3 using 256 word virtual buffers, the projected run time was 1:03, the actual was 0:57. For JNOD3, the projected run time was 2:28, and the actual 2:27. For WHIGE, the projected time was 7:22, and the actual turned out to be 7:25. We have been quite pleased with using virtual arrays, and have added the increased buffer capability to several other of our programs.

Many of the test runs gave us some ideas for possible future enhancement. We may well need larger disks to store more raw data, and RL01s or RL02s would seem to be both larger and faster than the RK05s. Winchester disks could provide large amounts of data storage at reasonable costs and would increase the speed to almost that of an RL01. If we decide to try to further decrease processing time in the future, there is no doubt that we will give serious consideration to buying some cache memory. If a processor upgrade is possible, the PDP 11/44 appears to be the best candidate.

It must be emphasized that these results are applicable only to our particular application programs. For "number crunching" activities a large computer with a floating point processor might be more attractive. In any case, we believe that the user is well-advised to perform a similar type of benchmarking to determine the best solution for his particular application.

DISTRIBUTION:

1121 L. W. Ebinger
1125 R. J. Longoria
1125 H. C. Ogden
1126 G. J. Hansen
1137 W. B. Pafford
1234 T. M. Unkelhaeuser
1241 D. E. Henry
1322 R. P. Kromer
1400 L. J. Heilman
1417 F. W. Muller
1420 R. H. Schultz
1421 W. J. Denison
1422 D. E. Murphy
1423 A. C. Littleford
1423 R. J. Isidoro (5)
1423 R. B. Ronan
1424 W. H. Carter
1424 L. G. Parsons
1424 R. Connell
1426 D. R. Kendall
1426 R. A. Newell
1426 L. J. O'Connell
1426 F. A. Ross
1426 A. J. Roth
1483 D. M. Morrison
1500 W. A. Gardner
1520 T. J. Hoban
1521 J. L. Mortley
1522 T. I. Barger
1523 E. A. Chipman
1523 B. T. Fox
1523 R. S. Frazer
1523 W. D. Love
1523 R. E. Trellue (25)
1523 W. H. Walker
1524 D. J. Miller
1542 D. E. Miller
1552 B. D. Hansche
1554 J. C. Bushnell
1554 B. Stiefeld
1738 N. A. Bourgeois
1763 R. C. Beckmann
1763 S. J. Weissman
2146 P. B. Bolwahnn
2313 R. W. Roberts
2313 C. M. Furry
2313 S. D. Nelson
2424 W. C. Burd
2551 R. B. Foster
2553 M. C. Jones
2563 J. M. Harris
2565 R. W. Barnard
2565 D. H. Jensen
2611 J. K. Wichelns
2611 R. K. Vokes
2614 G. C. Shepherd
2624 S. B. Gasser
2644 D. M. Darsey
2648 L. F. Tolendino
3313 G. E. Millard
3416 E. C. Domme
4252 E. L. Neau
4261 R. B. Spielman
5541 G. Jarrell
5636 J. K. Cole
5811 L. A. Harrah
5822 W. F. Chambers
8411 J. D. Benton
8452 D. N. Tanner
8214 M. A. Pound
3141 L. J. Erickson (5)
3151 W. L. Garner (3)
3154-3 C. H. Dalin (25)
For DOE/TIC (Unlimited Release)