CLIO/S Performance Considerations

August 17, 1998

Joe Casey
Phone: 914-433-7796
Tie line: 8-293-7796
email: joecasey@vnet.ibm.com
This document is available from the CLIO/S Worldwide Web page:
http://www.rs6000.ibm.com/software/sp_products/clio.html

Preface

This document is intended to serve two purposes. First, to demonstrate to prospective customers the potential of the fast, efficient interfaces that CLIO/S offers, while properly setting expectations in the context of total system performance. And second, to help users and system administrators arrange their resources and processes so as to make the most of the CLIO/S potential.

Performance and throughput are complex problems. There are no hard and fast rules. This document points out factors that are important in CLIO/S performance; however, it is not possible to consider all cases, or all the interactions that may occur. The intent here is simply to provide a framework and some examples for you to work with. You must determine how each factor applies to your application and its environment, using both analysis and experimentation.

IBM does not guarantee a particular level of performance in your environment.

No performance testers were injured in the production of this document.

© Copyright IBM Corp. 1995,1997

Contents

Preface . . . iii

Chapter 1. General Information . . . 1
Product information . . . 1
Terminology . . . 1

Chapter 1. CLIO/S performance in a nutshell . . . 3

Chapter 2. The Performance Task . . . 5

Chapter 3. Maximum Performance . . . 7
Memory-to-memory . . . 7
Discussion . . . 7
Instruction Counts . . . 8
Two terabyte database backup . . . 8
FBAR . . . 9
MVS CPU Utilization . . . 9
Parallelism and scaling . . . 9

Chapter 4. Customer Performance Experience . . . 11
Comments on customer experiences . . . 11
Migrate MVS DB2 to Oracle on distributed Sun workstations . . . 11
Environment . . . 11
Process . . . 11
Results . . . 11
Migrate MVS IMS to Oracle on an SP . . . 11
Environment . . . 12
Process . . . 12
Results . . . 12
Download MVS data to AIX DAT tape for distribution . . . 12
Environment . . . 12
Process . . . 12
Results . . . 12
Download MVS Seismic data to AIX for analysis, return output data . . . 13
Environment . . . 13
Process . . . 13
Results . . . 13
Backup SP nodes to MVS tape . . . 13
Environment . . . 13
Process . . . 13
Results . . . 14
A Situation where CLIO/S doesn't help . . . 14
Environment . . . 14
Desired process . . . 14
Discussion . . . 14

Chapter 5. Bottlenecks . . . 15
Looking for bottlenecks . . . 15
Single-process bottlenecks . . . 15
Test methods . . . 16

© Copyright IBM Corp. 1995,1997

Component performance table 16
Multiple-process bottlenecks 17
Resolving bottlenecks 18

Chapter 6. Hints, tips and notes 19
End user factors 19
Application factors 20
System factors 21
Support Levels 23
Number of Adapters - Striping 23
ESCON Director 23
AIX level 23
System Workload and Processor Speed 23
TCP/IP Performance and SP Tuning 24

Appendix A. Performance test tools 25
Using the Fake I/O function of CLFTP 25
CLFTP and CLPLINK timing statistics 26
IEBGENER job 27
perfwrit and perfread 28

Appendix B. Handy conversion chart, included free of charge 31

Chapter 1. General Information

Product information

CLIO/S is short for Client Input Output/Sockets Licensed Program, program number 5648-129. Version 2 Release 2 is now available, and this document is written with that version in mind.

The CLIO/S General Information Manual is GC23-3879. It is also available as CLIOGI PACKAGE from the IBM MKTTOOLS disk, or via anonymous FTP server: connect to lscftp.kgn.com, login with userid anonymous and your internet address as password. CLIO/S documentation is in /pub/pps/clios. The CLIO/S User's Guide is supplied as softcopy with the product.

Terminology

In this document:

'SP' refers to the RS/6000 SP.

One megabyte is 1,048,576 bytes.

One gigabyte is 1024 megabytes.

© Copyright IBM Corp. 1995,1997

Chapter 1. CLIO/S performance in a nutshell

- CLIO/S should be considered as one part of a total process performance solution
- Total process performance requires planning and flexibility
- Parallelism is the easiest way to achieve high performance
- With planning and flexibility, in almost all cases, CLIO/S will not be a bottleneck - it will move data as fast as you need.

© Copyright IBM Corp. 1995,1997

Chapter 2. The Performance Task

Creating an effective data transfer process requires some effort. Here is one view of the steps involved. This will be an iterative process, as the outcome of later steps may cause you to revisit earlier ones.

1. Determine your requirements
   - source and target
   - transfer size
   - transfer window
2. Design a transfer process
   - data extraction process
   - data transfer method
   - intermediate storage media
   - final storage media
3. Match the process to your system
   - Analyze components
     - capacity
     - workload
     - configuration
   - Step-by-step testing
     - identify potential bottlenecks
4. Implement the process
   - Install any new components
   - Write code, jcl, shell scripts, etc.
5. System test the process
   - Throughput
   - Resource utilization
   - Robustness
   - Ease of use
   - Total effectiveness

This document focuses on step 3.

© Copyright IBM Corp. 1995,1997

Chapter 3. Maximum Performance

This section discusses the maximum performance that can be achieved by CLIO/S. In actual practice, lower levels of performance will be seen.

Memory-to-memory

- Performance testing consisted of sending blocks of data many times and capturing average performance. Block sizes range form 60k to 1200k. Iterations vary between 90 and 1000. On systems with no other workload there was very little variation between runs.
- CPU utilization was gathered by various methods, including the AIX `iostat` command, TSO RMFMON, and the C language `clock()` function. Due to differences in these tools and the way they are used, the utilization percentages should only be considered to be rough estimates.
- Data in the table is for memory-to-memory transfer. Using the CLIO/S FTP or Tape Server interfaces will involve additional device overhead and will result in lower transfer rates.

  TCP/IP comparison

  This section used to contain a comparison between TCP/IP and CLIO/S. TCP/IP performance has improved substantially and the data was obsolete. This section will be updated with a new comparison in the near future.

- MVS system is a 600-J.
- AIX system is SP wide node.
- Our MVS test system ESCON channels run at 10 megabytes per second. 17 megabyte per second channels are available on larger processors.
- One megabyte = 1,048,576 bytes

Table 1. CLIO/S Performance

Channel Type	Workstation model	Direction	Megabytes per sec	CLIO interface	MVS CPU Utilization	RISC CPU Utilization
Protocol: CLIO/S CTC connection over ESCON						
ESCON	SP	from MVS	9.1	DPI	3%	9%
ESCON	SP	to MVS	8.3	DPI	3%	9%

Discussion

As advertised, CLIO/S can deliver 'near-channel' speeds, surpassing 80-90% of a 10-megabyte ESCON channel. By adjusting the transmitted block sizes, it is possible to surpass 98% of channel speed from MVS to AIX.

© Copyright IBM Corp. 1995,1997

Instruction Counts

Using the Generalized Trace Facility (GTF) on MVS, it has been observed that CLIO/S (and the associated MVS activity) uses less than 2000 instructions to accomplish a 4 kilobyte memory-to-memory transfer to AIX. This is in rough agreement with the utilization and transfer rate data measured in the above testing.

Two terabyte database backup

Here is a summary of a 2TB backup that was performed in the SP benchmark center.

Environment

The SP system that was backed up consisted of 64 thin3 and 4 SMP nodes. The total amount of data backed up was 2.002TB (or 2050.1GB). Except for 72GB of data, all of this disk resided only on 32 nodes so the backup was really performed on just these 32 nodes and the other 36 nodes were just backed up in small increments at the same time on the side. Transfer to MVS used 4 ESCON channels that were in the SMP nodes that were not part of the database.

Process

Data was read from disk via the `dd` commands and piped into `compress` processes whose output was piped into clplink processes which moved the data across the SP switch, up to MVS, and wrote it out to MVS disk.

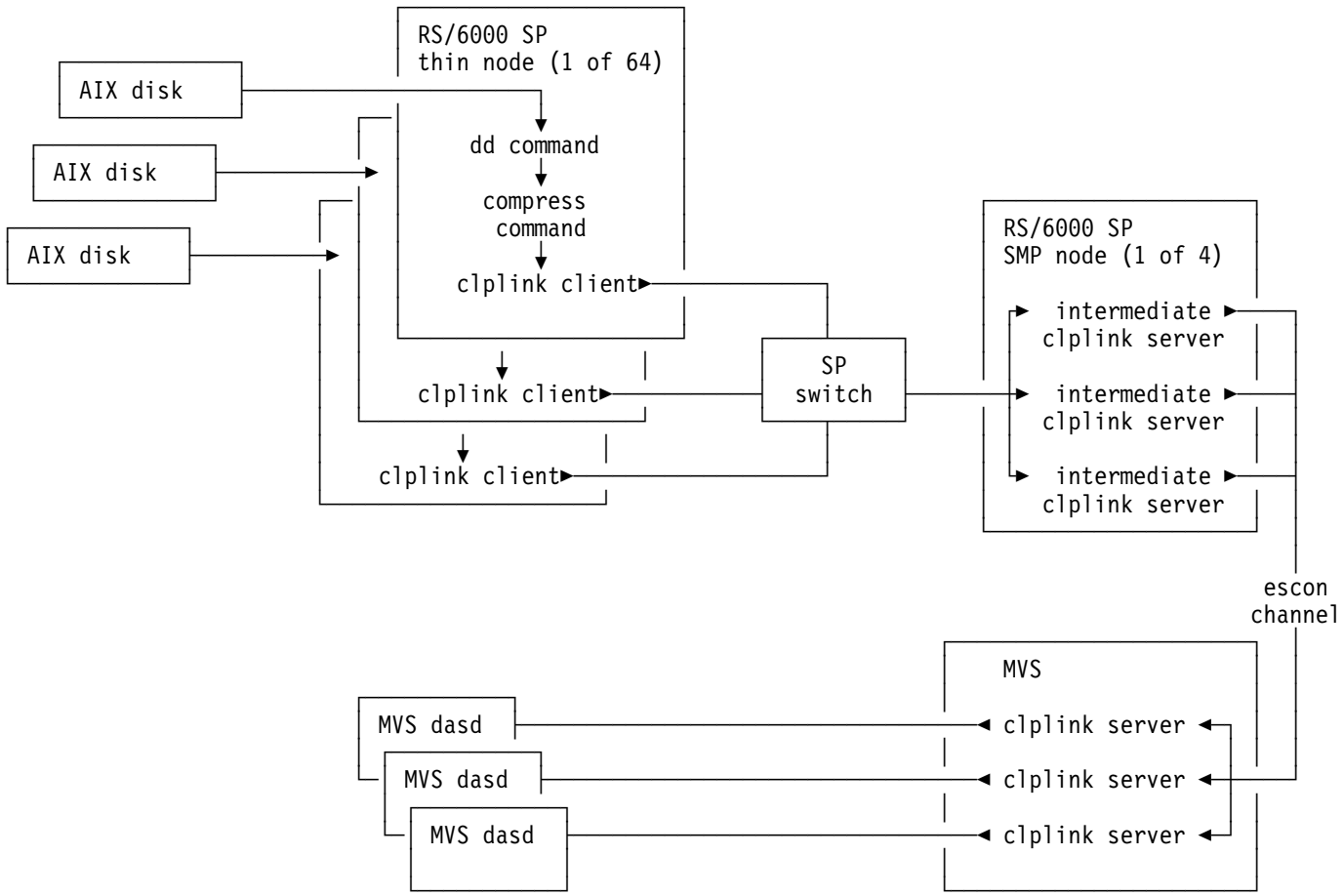

Figure 1. High speed disk to disk transfer

Results

It took approximately 8 hours 54 minutes from the start of the first job to the end of the last. This is approximately 224 gigabytes per hour or 62 megabytes per second.

The compress processes, running at about 1 megabyte per second, were the bottleneck in this environment. The complete backup could have been speeded up significantly by running more simultaneous dd->compress->clplink processes.

FBAR

FBAR is a high speed data mover that uses the CLIO/S APIs to load data from MVS to an SP used by the IBM SP benchmark center. It is a specialized internal utility not available to customers. It has achieved disk-to-disk transfers at a rate of 125 gigabytes per hour. To accomplish this, FBAR used 4 ESCON-attached SP wide nodes, each distributing data across the SP switch to 4 other nodes, where the data was written to disk. Using dasd striping on MVS and multiple processes, FBAR fed each ESCON connection between 8.5 and 9.2 megabytes per second over our 10-megabyte channels.

MVS CPU Utilization

Using some of the tools described later in this document, we have measured MVS cpu utilization when transferring tape or dasd files to or from AIX. In general, 75% to 90% of the total CPU usage is spent reading from or writing to the media, with CLIO/S using the rest to move the data across the ESCON (or block mux) channel.

Parallelism and scaling

CLIO/S is ideally suited for running multiple processes in parallel and scaling processes to match varying requirements.

- Performance limitations of a single process can often be overlooked if enough processes can operate simultaneously to achieve the total performance goal.
- Each CLIO/S process is separate - there is no central process that can become a bottleneck. On MVS, each CLIO/S process runs in its own address space.
- The number of physical components of a CLIO/S network can be scaled. This includes channel adapter cards, AIX nodes, MVS channels, MVS devices, and so forth.
- Slower components can be replaced with faster ones without substantial disruption of the process. For example, BMPX channels could be replaced with ESCON channels, or faster tape drives could be used.

Chapter 4. Customer Performance Experience

Comments on customer experiences

Getting maximum performance from CLIO/S requires a significant effort to analyze and configure the systems involved and tune the processes that use the CLIO/S interfaces. In general, our customers have tried to use CLIO/S with as little disruption of their existing configurations and practices as possible. This makes getting into production much easier, but limits the results that CLIO/S can achieve. Even so, our customers find that the high data rates and low overhead of CLIO/S allow them to do work that otherwise would not be possible.

The following sections describe some varied customer experiences with CLIO/S.

Migrate MVS DB2 to Oracle on distributed Sun workstations

Environment

- Amdahl processor
- 17 megabyte ESCON channel on wide node
- RS6K R24
- Other Unix boxes

Process

1. Extract MVS DB2 data to dasd
2. Use CLIO/S clplink processes
3. Feed data to AIX pipes
4. Direct AIX pipes to other UNIX boxes via SCSI pseudo-tape interface

Results

1. Sustained data rate of 14 megabytes/second for multiple processes
2. 50 gigabytes per night, 350 gigabytes per weekend
3. 250 CLIO/S jobs per night
4. Automated environment

Migrate MVS IMS to Oracle on an SP

© Copyright IBM Corp. 1995,1997

Environment

- MVS 3480 tapes
- ESCON channel on wide node
- SP with switch
- AIX SCSI and serial disks

Process

1. Extract MVS IMS data to MVS tape
2. Use CLIO FTP command
   - Read tape data
   - transmit via ESCON to SP
   - retransmit across switch
   - write to disk on SP nodes
3. Load files to Oracle database

Results

1. Sustained data rate of 3.5 megabytes/second for a single process
2. With multiple processes, 111 gigabytes in 7 hours, including manual tape mounts, etc.

Download MVS data to AIX DAT tape for distribution

Environment

- MVS proprietary database
- Blkmux channel
- 370 RISC machine
- SCSI DAT tapes

Process

1. C application extracts MVS data from dasd
2. Using CLIO/S tape interface, write data to AIX DAT tapes via remote tape server

Results

Since this customer's AIX tape drives are relatively slow, the transfer rate required is not too high. However, TCP/IP was not able to handle it. Customer's second solution was a vendor channel-to-scsi connector, but persistent reliability problems demanded a different approach.

With CLIO/S, customer feeds 8 scsi drives off 2 scsi controllers at a total transmission rate of 400-500 kilobytes/second.

The CLIO/S application was then easily ported to an all-AIX environment for workstation-workstation transfers.

Download MVS Seismic data to AIX for analysis, return output data

Environment

- 3490-E tape units, ESCON attached
- 8-way ES9000
- ESCON channels, 1 per SP node
- 16-node SP

Process

1. Proprietary code
   a. Read uncompressed seismic data from MVS tapes
   b. Using CLIO/S socket interface, distribute data to proprietary seismic analysis code on SP nodes
   c. Return output via CLIO/S DPI for MVS storage
2. Multiple concurrent processes
3. 24 hours a day, 7 days a week. Several terabytes of data per week.

Results

Since each process depends on a single stream of tape data, tapes are the unavoidable bottleneck in this environment. The requirement is for the proprietary code on both sides, including the imbedded CLIO/S socket calls, to be as about as fast as the tapes. Customer reports delivery of MVS tape data to SP nodes at 1.5 megabytes per second per process, 80-90% of the maximum possible in their environment.

Backup SP nodes to MVS tape

Environment

- MVS
- ESCON channels on 1 wide node
- SP machines

Process

1. Use NFS to gather data from remote nodes
2. Use CLIO FTP to move data to MVS
3. Write data
   - tape
   - dasd

Results

Data rate when writing to tape is 7 gigabytes/hour. Data rate when writing to dasd is 15 gigabytes/hour. This strongly indicates a slow tape system. Further analysis finds slow tape drives and channel contention. Resolution: more tapes and/or faster tapes and/or more channels.

Note that using CLIO/S FTP intermediate nodes would eliminate the need for NFS, probably improving performance.

A Situation where CLIO/S doesn't help

Environment

- 9 megabyte/per second tape drive on MVS
- Distributed database on RS/6000 SP
- ESCON connections

Desired process

- Back up database by using tar on entire logical volumes
- Use CLIO/S clplink to transfer tar files to MVS tape

Discussion

The fact that the MVS tape drive is inherently single-threaded makes the desired process too slow. Tar will only run at a few megabytes per second, and only one stream at a time can be written to MVS tape, so there is no benefit from a 9-megabyte tape drive, or a 17-megabyte/second ESCON channel, or CLIO/S. Writing to multiple temporary disk files on MVS is not acceptable due to the large size of the database.

To achieve high data transfer rates, some method of parallelizing this process must be found. In particular, adding additional tape drives (even slower ones) seems to be required.

Chapter 5. Bottlenecks

Looking for bottlenecks

Single-process bottlenecks

Any part of the total transfer path could be a performance bottleneck. In this section we will examine the components of a transfer path, indicating useful tools and analyzing how the performance of each component affects the whole.

The path we will work with in this section is as follows: from a disk drive on an SP node, to the node, through the SP switch to an ESCON-attached node, through the ESCON connection to an MVS system, to MVS dasd.

This graphic shows individual components of performance that can be used to help analyze total system performance. The numbers associated with each item refer to the entries in the table below, where specific results are shown for the CLIO/S test environment.

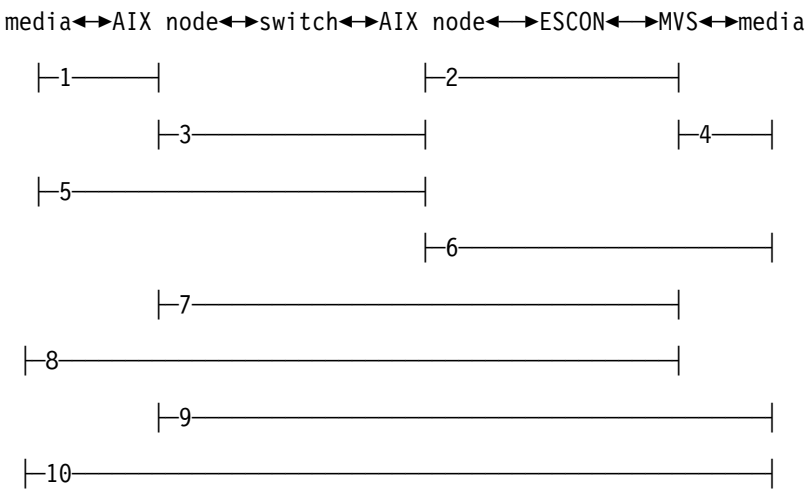

Figure 2. Components of a complete data transmission path

Note that we could break the system into even smaller parts. For example, MVS dasd performance is affected by number and configuration of channels and control units. However, for this report this level of detail is ignored, and the term 'dasd' encompasses everything involved in getting disk storage into memory.

The following diagram shows the particular test environment used for this report.

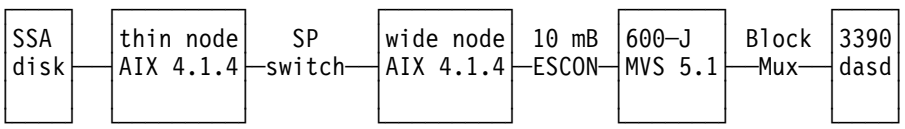

Figure 3. Test configuration

© Copyright IBM Corp. 1995,1997

Notes:

- Our MVS system is a virtual=real guest on a VM system.
- Our ESCON channels run at 10 megabytes/second. 17 megabytes/sec channels are also available.
- MVS dasd units are connected to MVS via block multiplex cables.

Test methods

The following test methods were used to gather the data reported here.

1. perfwrit and perfread, extremely simple c programs. See Appendix A, “Performance test tools” for more information.
2. FCFIDUM. This is a CLIO/S DPI application distributed as sample code with CLIO/S.
3. IEBGENER job. See Appendix A, “Performance test tools” for more information.
4. clftp fake I/O. See Appendix A, “Performance test tools” for more information.
5. clftp

The test methods were chosen for ease of use and availability, and are not necessarily capable of achieving the highest possible performance. Thus the results reported for each component should be considered as lower bounds.

Component performance table

The following table shows the results we have seen for the particular components in our environment. Your results may differ.

The 'test method' column refers to one of the test methods mentioned in the previous section.

The 'AIX > MVS' and 'AIX < MVS' entries in the performance columns refer to the direction of transfer in accordance with Figure 2, even if the path component does not go all the way from one system to the other.

Table 2. Transfer path component performance

Tag	Path component	Test method	Performance (megabytes/sec)	
			AIX > MVS	AIX < MVS
1	local disk <> local node	1	4.8	4.8
2	intermediate node <> MVS	4	7.8	7.8
3	local node <> intermediate node	2	13	13
4	MVS <> MVS disk	4,3	2.9	3.4
5	local disk <> intermediate node	4	4.6	5.2
6	intermediate node <> MVS disk	4	3.3	3.5
7	local node <> MVS	4	4.7	7.6
8	local disk <> MVS	4	4.1	6.0
9	local node <> MVS disk	4	2.8	3.5
10	local disk <> MVS disk	5	2.8	3.5

Comments on the table

In performance, the whole is always somewhat less than the sum of the parts. The total process will never be faster than the slowest component, and in practice will be somewhat slower.

However, the table shows that CLIO/S is not likely to be the slowest component, and will not create a new bottleneck. This suggests that to get the most value from CLIO/S and the channel-to-channel connections, multiple concurrent processes should be used.

Multiple-process bottlenecks

Whenever resources are shared in a multiple-process environment, new bottlenecks can occur. Some likely candidates:

- RS6K cpu, when processes are on a single node
- ESCON or block mux capacity, when processes share a CTC connection
- AIX disk utilization, when processes read/write from the same disk
- MVS channel utilization, when read/write devices share paths

Resolving bottlenecks

Bottlenecks are resolved by either increasing the capacity (throughput rate) of the bottlenecks or by bypassing them somehow.

Increasing the capacity of a particular path component could involve replacing it with a faster one, providing a faster connection to it, providing more connections to it, or doing some performance tuning.

Bypassing a bottleneck could be accomplished by reengineering the process to eliminate a step. For example, transferring data generated by an AIX application directly into a CLIO/S pipe for immediate transfer to MVS for storage, rather than staging the data to AIX files for subsequent transfer.

In other words, there are three main approaches to resolving bottlenecks:

- Change the system
  - Add more/faster ESCON adapters, tape drives, nodes, etc.
  - Reconfigure channels, control units, job priorities, etc.
- Change your process
  - Change process scheduling to minimize contention
  - Make processes more efficient - move less data
- Change your expectations. Learn to live within your limitations

Any of the methods discussed in Chapter 6, “Hints, tips and notes” could be useful in resolving bottlenecks.

Chapter 6. Hints, tips and notes

Earlier sections of this document have attempted to take a total system view of performance. This section looks at isolated factors that may be helpful (or harmful). Many of these items have already appeared in the earlier discussions.

The factors are discussed in groups:

- End user factors
- Application factors
- System factors

End user factors

Factors that can be controlled by the end user include:

- Concurrent processes
- Choice of CLIO/S interface
- Compression
- Media contention
- Disk fragmentation
- No channel-to-channel connection

Concurrent Processes

CLIO/S I/O operations proceed in a step-by-step manner. Inevitably, some system resources will be idle while CLIO/S is using others. To get the most out of all resources, you can create several concurrent CLIO/S operations.

There are several ways this can be accomplished.

- Multiple processes using CLIO/S. This will occur naturally in many production environments.
- Using the CLIO/S stripes function. See the CLIO/S Users Guide and Reference for details.

Choice of CLIO/S interface

The CLIO/S tape interface used by the tape pseudo device driver processes records one at a time. When writing, each record is completed before the next is sent. When reading, there is a read-ahead that allows one record to be read in advance. On MVS the maximum record length is 32k. Compared to the clplink or clftp interfaces, which send buffers of several hundred thousand bytes, and use multiple buffers to overlap processing, this 32k limit degrades performance.

If the tape device is slow, or the process that reads or writes the tape is slow, this is not a big problem. But for large, high-speed transfers, clplink or clftp will be faster.

Note that clplink and clftp can read and write MVS tapes, and in many cases it is possible to adapt a process that uses a tape device (such as a tape pseudo device driver) to instead use a named pipe, in which case clplink can be used.

© Copyright IBM Corp. 1995,1997

Compression

In a single-process environment data compression is generally not useful. This is because the compression routines are relatively slow compared to CLIO/S transfer speeds, so compression becomes the bottleneck all by itself. Of course, if the data were already compressed before the transfer began, that would improve the transfer rate accordingly.

Data compression may be helpful in a multi-process environment. Consider the case where many SP nodes are feeding data to a single ESCON-attached node for transfer to MVS. If the ESCON-attached node becomes CPU-bound, or the ESCON channel approaches full utilization, it might be useful to compress the data at the originating nodes, decreasing the demands on the ESCON node and channel.

The speed of the compression routine and how well compression can be integrated into the total process are important factors.

Avoid media contention

Avoid having multiple processes that read or write using the same device or channel path.

Avoid disk fragmentation

This concept has somewhat different meanings for AIX and MVS, but the general idea is the same - spreading a single file across many small sections of a disk hurts performance.

No channel-to-channel connection

In general, CLIO/S will attempt to establish a channel-to-channel connection between AIX and MVS systems. If the connection is not established, the default behavior is to use TCP/IP instead. In some cases this is desirable, but it may be that you would rather have the process terminate if it is not able to achieve high performance. If the CLIO/S environment variable CLIOFASTIO is set, the process will terminate if no channel-to-channel connection is achieved. You must have up-to-date CLIO/S maintenance for this function to work.

Application factors

Along with the command line interfaces of CLIO/S, customer-written applications can be developed using the DPI, CMI, or socket interfaces. Factors that can be controlled by the application writer include:

- Size of the transmit/receive buffer
- Alignment of the transmit/receive buffer
- Concurrent processes

Size of the Transmit/Receive Buffer

CLIO/S sends data in 60K chunks. Your application gets best performance when it transmits data in blocks that are a multiple of this size. In general, the larger the transmission size, the better.

Alignment of the Transmit/Receive Buffer

If possible, use 4k page boundaries on AIX for the data you transmit via CLIO/S. If your data is not aligned, the adapter device driver must copy it to 4K-aligned storage. In our testing, using aligned storage improves transmission rates by a few percent, and more importantly cuts AIX cpu utilization roughly in half.

Concurrent processes

The application developer has more ways of obtaining concurrency, such as forking multiple processes or using multiple buffers for I/O. These should be considered in the design process.

System factors

Factors that are controlled by systems administrators and/or hardware planners include:

- AIX system memory
- Supplying sufficient subchannels
- Configuration of the adapter
- Support levels
- Number of adapters
- ESCON Director
- System workload and processor speed
- TCP/IP performance and SP tuning

AIX system memory

When an application such as CLIO/S writes a file to disk, AIX will in fact hold as much of the data as possible in memory, doing actual disk I/O only when necessary. This can be very beneficial for performance, but it complicates the analysis of performance problems, and it is necessary to understand what is going on.

Here is an example from our own testing, using the `perfwrit` tool described in Appendix A, “Performance test tools” to write a 100,000,000 byte file to AIX disk.

```
# rmss -p
Simulated memory size is 512.00 Mb.
# vmstat
procs    memory              page              faults        cpu
----- ----------- ---------------------- ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 10976 124466   0   0   0   0   65   0 112   37  24  0  2 97  0
# time perfwrit testfile

real    0m2.12s
user    0m0.10s
sys     0m2.00s
# ls -l testfile
-rw-r--r--   1 root     usr      100000000 Jun 22 13:23 testfile
```

The performance (47 megabytes/sec) seems excellent, but this is deceptive, because the data was not actually written to disk, but rather kept in memory. Note the 124466 free pages shown by `vmstat`.

Now the test is repeated after using the rmss utility to limit the amount of memory available. See *SC23-2365, Performance Monitoring and Tuning Guide*, for more on rmss. rmss is available as part of the Performance Toolbox on AIX 4.x systems.

Important note!

After using rmss, dont forget to issue rmss -r to reset your system memory to the proper value.

```
# rmss -c 32
Simulated memory size changed to  32.00 Mb.
# vmstat
procs    memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0 10976   238   0   0   0   0   65   0 112   38  24  0  2 97  0
# time perfwrit testfile

real    0m35.22s
user    0m0.02s
sys     0m3.31s
```

The transfer rate (2.8 megabytes/sec) is much slower, because now all the data is actually being written to disk.

The benefit of extra memory is real, but it should not be mistaken for fast device I/O.

All of the file I/O done for tests shown in this document is 'real'.

Supplying sufficient subchannels

To allow additional CLIO/S processes to run simultaneously, there must be sufficient subchannels (device addresses in MVS terminology) to support them. Each CLIO/S process requires its own subchannel. See the CLIO/S User's Guide and Reference for details on defining subchannels.

Configuration of the Adapter

Another configuration variable is the number of buffers allocated to transmit (from workstation to the mainframe) and receive (by workstation from mainframe).

The PCA adapter uses 52 4K buffers, 26 for transmit and 26 for receive. You can improve transmit performance without impacting receive performance by increasing the number of buffers allocated to transmission. Try 36 transmit buffers and 16 receive buffers as a possible starting point. You can make this change using either the SMIT menu interface or the AIX **chdev** command.

This example shows how to use the **chdev** command to reallocate the buffers:

```
su root
chdev -l cat0 -a xmitno=36 -a recvno=16 &
```

The ESCON adapter dynamically allocates its buffers. You should not attempt to tune it.

Support Levels

You **must** have the correct level of device driver and adapter microcode to use CLIO/S. Having back-level microcode or device driver code can cause poor performance or outright failure, especially as more data is driven through the adapter.

See **CLIO/S and Related Software Maintenance** for details about the latest device driver, microcode, and CLIO/S maintenance.

Number of Adapters - Striping

CLIO/S can use more than one channel adapter. If ESCON channel utilization is the problems, consider adding additional adapters and using more chpids for CLIO/S.

A single CLIO/S process can use more than one adapter ('striping'). However, this does not provide any performance benefit unless the process is driving data faster than a single adapter can handle it, which is unusual.

Multiple processes will benefit from multiple adapters whenver the total throughput of all the processes nears or exceeds the capacity of a single adapter.

CLIO/S is capable of striping multiple subchannels on a single adapter, but there is no performance benefit in doing so.

ESCON Director

Our testing shows that the ESCON director does not degrade CLIO/S performance, even when switching between two different paths carrying CLIO/S traffic.

Of course, if the chpid used for CLIO/S CTC connections is also shared across other devices, that limits the total bandwidth available to CLIO/S processes.

AIX level

Our testing shows no difference in performance between CLIO/S performance on an AIX 4.1.x system as compared to an AIX 3.2.x system, all other things being equal. Most of the AIX data for this document was gathered on an AIX 4.1.3 system.

System Workload and Processor Speed

While efficient, CLIO/S still draws on system resources, and competition for resources among the various processes in the system has an effect on CLIO/S performance and throughput. On the RISC/6000, high transmission rates will result in high CPU use, with the exact ratio depending on the particular machine in use. If multiple adapters are used, it is possible for CPU use to be a gating factor.

AIX commands such as `vmstat 5` can be used to observe CPU utilization.

Mainframe use will typically be only a fraction of that seen on the workstation, so mainframe processor speed is not a key consideration.

TCP/IP Performance and SP Tuning

All CLIO/S interfaces use TCP/IP for initialization. However, the CLIO/S Sockets interface uses TCP/IP as part of each `send` or `recv` call. Poor TCP/IP performance, especially high latency, has a noticeable impact on the performance of the CLIO/S Sockets.

Concurrent processes can minimize the impact of high latency.

Specific tuning paramters for the SP switch include setting `tcp_mssdflt` to 4046, and setting the ESCON adapter TCP/IP interface MTU to 4096.

The SP System Administration Guide, GC23-3897, has section 8.1.3.3.1, 'Tuning for User Environments'. The recommendations there are very useful. See the table entries for 'Scientific/Engineering High Performance Large Data Streaming'. Some of the recommendations will also be applicable to non-SP networks of RS/6000 workstations.

TCP/IP tunable settings, such as `tcp_sendspace` and `tcp_recvspace`, should be the same for any two nodes communicating with each other.

Appendix A. Performance test tools

Using the Fake I/O function of CLFTP

The clftp interface of CLIO/S offers a 'fake I/O' function that is useful for evaluating the performance of individual components of a CLIO/S network. The fake I/O function is controlled by the environment variables CLFTPFAKEIOREM and CLFTPFAKEIOLOC.

Important note!

If you use these environment variables for performance analysis, remember to 'unset' them when you are finished, or your subsequent clftp transfers will fail.

Using CLFTPFAKEIOREM

Set CLFTPFAKEIOREM (the 'REM' is for remote) to some positive integer before starting your clftp session. For example, `export CLFTPFAKEIOREM=104857600`. After starting the clftp session, set the clftp transfer Mode to 'binary'. Then clftp `get` commands will result in the remote clftp server sending 104,857,600 bytes (100 megabytes) of arbitrary data, without doing any disk or tape I/O on the remote system. On the local system, the data will be written to the file named by the get command as usual.

Likewise, the clftp `put` command will cause the remote clftp server to receive 104,857,600 bytes of data but not write it to tape or dasd.

The purpose of this function is to isolate the aspects of performance involved with: 1. writing or reading from dasd or tape on the local system, and 2. sending the data between systems (possibly a sequence of systems) via CLIO/S connections, while ignoring the effort required to read or write the data on remote media.

Using CLFTPFAKEIOLOC

For corresponding function on the local side, set CLFTPFAKEIOLOC (the 'LOC' is for local) to some positive integer before starting your clftp session. For example, `export CLFTPFAKEIOLOC=104857600`. After starting the clftp session, set the clftp transfer Mode to 'binary'. Then clftp `get` commands will result in the remote clftp server sending the file named in the `get` command (up to 104,857,600 bytes) without doing any disk or tape I/O on the local system. Likewise, the clftp `put` command will cause the local clftp server to send 104,857,600 bytes (100 megabytes) of arbitrary data, which the remote server will write to media as specified by the `put` command.

The purpose of this function is to isolate the aspects of performance involved with: 1. writing or reading from dasd or tape on the remote system, and 2. sending the data between systems via a CLIO/S connection, while ignoring the effort required to read or write the data on local media.

Here is an example of CLFTPFAKEIOLOC usage.

© Copyright IBM Corp. 1995,1997

```
tserv8.ppd.pok.ibm.com: 14> export CLFTPFAKEIOLOC=104857600
tserv8.ppd.pok.ibm.com: 15> clftp kn1mvs
Connecting to : kn1mvs
Remote system kn1mvs job JCASEYO connected with protocol ESCON.
clftp > binary
200 File is IMAGE.
clftp > site dir=0 blocks pri=4000
200 Command completed successfully.
clftp > stat

Connected to: kn1mvs (MVS)
ascii: Off              Status Freq.: 4
Last Site String: dir=0 blocks pri=4000
Transport Protocol: ESCON
Translation Site: Local
Maxbuflen = 500000      Firstbuflen = 100000    Secondbuflen = 100000

 Remote file allocation values.

File Organization  : Physically Sequential.
Record Format      : Variable Blocked.
Record Length      : 260
Block Size         : 27648
Storage Unit Type  : Blocks.
Primary Units      : 4000
Secondary Units    : 1
Directory Blocks   : 0
Volume             :
Dcbdsn             :
Unit               :
Alloc              :

200 Command completed successfully.
clftp > put dummy 'jcasey.hundred.megabyte'
226 Transfer complete(104,857,600 bytes).
    Elapsed time: 34.77 sec.     2.88 Meg/s
```

The 2.88 megabyte per second transfer rate reflects the speed that clftp can write to MVS 3390 dasd in our environment.

Using Both

Both CLFTPFAKEIOREM and CLFTPFAKEIOLOC can be set to the same value. In this case, no media I/O is done on either side. The purpose of this function is to isolate the performance of clftp transfer over CLIO/S connections. Note that this performance will not be as good as the memory-to-memory transfers described in “Memory-to-memory,” since the clftp interface does extra handshaking and checking.

CLFTP and CLPLINK timing statistics

CLFTP and CLPLINK are both capable of generating very detailed timing statistics covering both sides of a file transfer. The problem with these statistics is that they are undocumented and there is only one person on earth who can read them. If you need help interpreting these, send them to us, and maybe we can induce the developer to create some documentation.

CLFTP timing statistics

From within a clftp session on AIX, issue the command `timing`, then perform your transfer. Detailed results will be found in `clftp.stats`.

CLPLINK timing statistics

Set the AIX environment variable `CLPLINKTIMING` (Don't set it to any particular value, just set it.) Then perform your clplink transfer. Detailed results will be found in `clftp.stats`.

IEBGENER job

The following IEBGENER job is a simple way to get an idea of how fast a dasd file can be read in your MVS system. Note that the resulting statistics will reflect system workload, channel contention, and similar variables apart from raw dasd speed. We could try to eliminate all those variables, but it would be a lot of work, and the end result would probably not be as useful as this 'real-life' method.

```
//DUMCOPY JOB (572260,6MM,202-3),'&SYSUID',
//            NOTIFY=&SYSUID,REGION=4096K,CLASS=A,
//            MSGCLASS=4,MSGLEVEL=(1,1)
//*-------------------------------------------------------------
//GENERAL  EXEC PGM=IEBGENER,TIME=1440,REGION=1048M,DYNAMNBR=80
//SYSPRINT DD  SYSOUT=*
//*SYSUT3   DD DISP=(,DELETE),SPACE=(CYL,(20,20))
//*SYSUT4   DD DISP=(,DELETE),SPACE=(CYL,(20,20))
//SYSUT1   DD DISP=SHR,DSN=JCASEY.HUNDRED.MEGABYTE
//SYSUT2   DD DUMMY,DISP=(,DELETE),UNIT=VIO,
//         SPACE=(TRK,(1400,20))
//SYSIN    DD  DUMMY
```

Figure 4. IEBGENR job

Check the SYSOUT from this job to find timing statistics. Here is an example from our test environment. We used the following 100 megabyte dataset, which was created using the clftp fake i/o function described above.

```
 Data Set Name  . . . . . : JCASEY.HUNDRED.MEGABYTE

 General Data                          Current Allocation
  Management class  . .. : STANDARD      Allocated blocks  . : 4,000
  Storage class   . . .. : BASE          Allocated extents   : 1
     Volume   . . . . .. : PRIMP3
     Device type  . . .. : 3390
  Data class  . . . . .. : DEFAULT     Current Utilization
     Organization . . .. : PS            Used blocks . . . . : 3,865
     Record format  . .. : VB            Used extents  . . . : 1
     Record length  . .. : 260
     Block size . . . .. : 27648
     1st extent blocks   : 4000
     Secondary blocks .  : 1
```

Our IEBGENER job output includes the following:

```
+-------------------------- I / O  ACTIVITY  TOTALS --------------------------+
|DASD=  3,866  TAPE=      0  VIO=      0  OTHER=      0  WITHOUT A DD=      7 |
+------------------------------------------------------------------------------+
IEF373I STEP /GENERAL / START 96073.1340
IEF374I STEP /GENERAL / STOP  96073.1341 CPU    0MIN 05.18SEC SRB    0MIN 00.67
+-|JOB  STATISTICS|-----------------------------------------------------4.1-+
|JOB DUMCOPY (JOB04149) ENDED: STEPS(  1) CLASS(A) PGN(    1) PRTY(  0)       |
+------------------------------------------------------------------------------+
|ELAPSED TIME: 00:00:29.25  CPU TIME(TCB): 00:00:05.18  (SRB): 00:00:00.67    |
|RCT CPU TIME: NONE         I/O INT. TIME: 00:00:00.09  HIPER: NONE           |
|SERVICE UNITS: CPU=   48,653  SRB=    6,137  I/O=   19,360  MSO=   80,829    |
|TRANSACTIONS: ENDED:NONE   ACTIVE FOR: 00:00:29.15 RESIDENT FOR: 00:00:29.15 |
```

Dividing the 100 megabytes by 29.15 seconds (active time) we get a disk read rate of 3.43 megabytes per second in our environment.

perfwrit and perfread

These are ridiculously simple routines that write and read a big AIX file.

perfwrit.c

Usage: `time perfwrit testfile`

```
/* Write a file for performance testing */
#include <stdio.h>
#include <fcntl.h>

main(int argc, char **argv)
{
   int     fd, rc, x, filesize = 104857600, block = 10240;
   char    *buf1;

   fd = open( argvff1“ ,O_RDWR | O_CREAT , 0666);
   buf1 = (char *)calloc(block,sizeof(char));
   for ( x=0; x < filesize/block; x++) rc = write(fd, buf1, block);
   exit(rc);
   }
```

Figure 5. perfwrit.c disk write tool

perfread.c

Usage: `time perfread testfile`

```
/* Read a file for performance testing */
#include <stdio.h>
#include <fcntl.h>

main(int argc, char **argv)
{
   int     fd, rc, x, filesize = 104857600, block = 10240;
   char    *buf1;

   fd = open( argvff1“ ,O_RDWR | O_CREAT , 0666);
   buf1 = (char *)calloc(block,sizeof(char));
   for ( x=0; x < filesize/block; x++) rc = read(fd, buf1, block);
   exit(rc);
   }
```

Figure 6. perfread.c disk read tool

Appendix B. Handy conversion chart, included free of charge

The chart is approximate, limited by the granularity of the font.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Megabytes per second
0	3	6	9	12	15	18	21	24	27	30	33	36	39	42	45	48	51	54	Gigabytes per hour
0		1		2		3		4		5		6		7		8		9	Terabytes per week

Figure 7. Transfer rate conversion chart

© Copyright IBM Corp. 1995,1997