SLAC-PUB-3565
January 1985
(N)

THERE IS NO I/O LIKE NO I/O *

T.Y. Johnston

Stanford Linear Accelerator Center
P. O. Box 4349
Stanford, California 94305

ABSTRACT

On most computer systems the most common cause of performance degradation is I/O contention. This paper will examine some efforts that can be taken in a VM environment to reduce I/O or its effect at both the global and local levels.

Invited talk presented at Candle Users Group,
Los Angeles, California, February 21 & 22, 1985

*Work supported by the Department of Energy, contract DE-AC03-76SF00515.

OVERVIEW

From the time that most systems are IPLed until they are shutdown almost all tasks spend most of their time waiting for I/O. Cray Research on the XMP has solved this problem for many applications with their gigabyte per second solid state device. IBM has hardly addressed this problem. When one looks at the improvements in CPU performance versus I/O service times over the last decade CPU performance has improved much more than has I/O service time. (As used here I/O service time is a combination of data transfer time, seek time, rotational latency, and interference.) Today's system is fundamentally more I/O bound than the one of a decade ago. I will discuss at three different levels some considerations for improving the I/O performance.

1. CP system wide considerations. What can be done to system I/O to improve performance?
2. A few things that can be done at the CMS level to improve performance.
3. Some considerations at the application level to reduce I/O.

SOME CONSIDERATIONS

YOU CAN'T TUNE IT, IF YOU CAN'T MEASURE IT!

VM, as delivered by IBM, comes with insufficient measurement tools. To be able to do even a reasonable job of tuning your system or an application you must obtain tools. There are now several vendors who are providing tools to assist in the task. My installation has found that we have to build many of our own tools in addition to using those supplied by vendors. Hopefully, this situation will improve in the future as more products come into the market place.

As an example of one need, when trying to tune I/O one of the most important data elements is the number of SIOs done by an operation. There is no standard CP command to give you this information, even though CP maintains SIO counts in both the RDEVBLOK and the VDEVBLOK. We have added query commands to CP to obtain this information. If you don't want to do a system modification, people with class C or E privileges can get this information by having an EXEC that uses the CP LOCATE and the CP DCP commands.

My discussion is oriented towards a medium to large system where there are 50 to 300 or more connected users. In this environment the system is the predominent cause of I/O on a volume. No user, with the exception of a data base manager, has any significant effect. Thus you are looking at CP paging, swapping, and directory space; CMS S and Y disk; and any special applications such as a data base manager. One can dedicate volumes for such space, but one should not reduce the number of arms servicing the data to get it on dedicated volumes. For example, if you use two dedicated volumes for CP paging on 2 channels you can support a paging rate of about 90 I/Os per second. On the other hand eight 200 cylinder areas on eight volumes on four channels can support about 360 I/Os per second.

SYSTEM WIDE CONSIDERATIONS

Some things to consider system wide:

1. If you have any solid state devices use them for paging.

2. Put your other high activity data areas on 3380s.

3. Spread high activity areas over channels and arms.

4. If using AA4 3380s spread activity over internal paths.

5. HPO 3.4 swapping should be spread over all channels. Any data co-resident on a volume with swapping should be very low activity (e.g. CP source). Don't put any user's 191 disk on a swapping volume.

6. The CP directory on many systems is a very busy data area. The directory is searched linearly. A bypass that helps is to put frequently linked disks such as the S and Y in a directory entry which comes at the front of the directory. For those that have a sorted directory (such as created by DIRMAINT) SLAC has created a mod (which is on the Waterloo tape) which finds any directory entry with one I/O.

CMS CONSIDERATIONS

Most of the considerations for CMS (without doing system modifications) are in the areas of disk blocking, minimizing seeks, and minimizing contention. Some things to look at:

1. Consider the recently announced IBM 3880 model 23 and supporting VM software which allows minidisks (such as the S and Y disks) to be cached. This can markedly improve S and Y disk performance.

2. Ensure that both the S and the Y disk are separate from other high utilization disks.

3. Ensure that all disks which have significant activity use 4K blocks. In fact it can be argued that all mini disks on 3380 should be blocked 4K. Occasionally a disk with lots of small files will require slightly more space at 4K but that is probably the five percentile case. If you use smaller blocks, every file that fits in one block at 4K and doesn't fit in one block on the smaller blocksize requires a minimum of three I/Os instead of one (2 or more for the data blocks and 1 for the index block).
4. Create the S and Y disks by copying all modules, execs, and XEDIT macros to the disk first (so as to clump them together and minimize seeks).
5. If you make changes to the S and Y disk without recreating them, then periodically rebuild the S and Y disk by copying to another address and copying back. If these disks are not rebuilt then the blocks of a file can be spread all over the disk. At SLAC we discovered that the Fortran compiler (the principal language at SLAC) was spread over 15 cylinders.
6. Wherever possible use DCSSs rather than modules for products such as VS FORTRAN. The paging system is more efficient than the CMS file system and only the required pages are read in rather than the entire module.
7. Put other local codes into DCSSs if possible. At SLAC we have a standard set of nucleus extensions for highly used CMS components that are NUCXLOADed for all virtual machines. These

reside in a DCSS allowing one copy in the paging space to satisfy the community.

8. Make certain that S and Y disks match the saved system so that people do not get private copies of the directories. If you follow a policy of updating either of these disks on the fly, then you need to establish procedures that keep the disks and saved systems in synch.

APPLICATION CONSIDERATIONS

Under CMS fixed length records can be handled much more efficiently than variable length records. Let me give an example: We have a query command that XEDITs a file in order to provide the command response. Some users were having trouble because of the size of the file. We reasoned that we could reduce the amount of memory required to edit the file by changing the lrecl from 80 to 25. When we did this, the file took the same amount of memory and the I/Os to read the 3500 records went from 6 to 23! We found that in the first case the default XEDIT WIDTH equalled the record length and consequently XEDIT read in 15 blocks at a time. In the second case only one block at a time was read in by XEDIT as it scattered the records through memory 80 bytes apart. The CMS file system allows the reading and writing of multiple fixed length records in a single call to the file system, allows you to read or write one record at a time, and requires you to read or write only one variable record at a time.

The following table summarizes various cases:

3554 Record File with Various Strategies

RECFM	LRECL	Number Blocks	Width	Number I/Os
FB	80	70	80	6
VB	80	69	80	70
FB	25	22	80	23
VB	25	22	25	23
FB	25	22	25	3

From this it can be seen that applications with records all about the same size get significantly better performance using fixed block records rather than variable. Make certain tht you specify a blocksize close to 32k (that value can go to 64K if you are not using OSSIM).

In the case of FORTRAN one should make sure that all large data files are written using unformatted I/O, mode 4, and close to 32K blocksize. Each FORTRAN 'block' appears to CMS as a variable length record of length blocksize. Thus CMS will read the record in one I/O even though it spans seven disk blocks (assuming the blocks are on the same cylinder).

Don't use EXECs to manipulate large amounts of data. EXECIO only reads one block at a time.

The interface between applications programs and the spooling system is also inefficient. Although, you will not save real I/Os you will save significant mumbers of virtual I/Os and significant CPU time if you write print files to disk (particularly as FB files) and then use the PRINT command to print them.

SUMMARY

I have given some hints and ideas about improving system performance by reducing I/O. The most important thing that anyone can do in their own installation to improve performance is to:

a. Install measurement tools in your system.

b. Learn how your system runs.

c. Improve it.

d. Measure to make sure that you did improve it.

e. Repeat the process.