VS037

Monday, May 9, 1994

2:30 p.m. – 3:30 p.m.

Intermediate/Technical

OpenVMS System Performance Management II: Disk and File Management

Harry E. Flowers

The University of Memphis

112 Administration Bldg.

Memphis, TN 38152

flowers@narnia.memphis.edu

Disclaimer: The author and The University of Memphis accept no responsibility for the use or reliabilty of information presented in this session.

Disk and File Management

- Maintaining free space
- Fragmentation – identifying and correcting
- I/O balancing – disks, controllers, and striping
- RMS indexed file tuning and global buffers

Maintaining free disk space

Why free space? As a disk fills, applications suffer.

- The file system will have to try harder and harder to find available clusters in the bitmap. Once the cache is exhausted, searching through the bitmap for scarce free blocks is *very* time consuming.
- Contiguous space will be far scarcer than free space. This impacts extending directories and the index; file creations will generally fail before you fill the disk if many files are involved.
- Disk defragmenters are unable to cope with a full disk or a fragmented index; BACKUP is the only way to defragment such a disk.
- Performance degradations can easily be a factor of ten or more once you get 92% to 96% full. It just gets worse from there.

Maintaining free disk space, cont'd.

Some general principals about disk space:

- User disks should be considered full at 70% – 80% of the actual disk blocks, depending on applications.
- Highly dynamic disks (such as scratch devices) should average far lower to have space available for high activity times.
- Static disks (for example, a disk with only page files on it) may be practically full with no problems.
- Plan for your free space as well as file system overhead when you're planning your disk purchases. Planning for a 1GB disk drive when you need 1GB of data space is foolish.

The system disk

Bad things can happen when you fill the system disk. For example, if you don't want the audit server to hang the system when the disk space approaches it's threshold, you might wish to use the `SET AUDIT` command to change this behavior. Since the `SET AUDIT` command varies depending on your version of OpenVMS VAX or AXP, you'll need to look up the correct commands to use for your version.

You can check the current setting with the `SHOW AUDIT/ALL` command. Also, remember that spooled devices default to the system drive, so large print files sent to the print device instead of the queue may have a significant impact. You might wish to have them spooled to a scratch drive instead.

The system disk, cont'd.

There are several files which reside on the system disk by default that just keep growing. OpenVMS upgrades usually clear some of them, but only in the system root performing the upgrade.

SYS$MANAGER:OPERATOR.LOG The messages that go to the operator console also go to this file. A new file is created each time the system is booted. You can also cause a new file to be created with the `REPLY/LOG` command. You should purge these files regularly. [This is a text file.]

SYS$MANAGER:SECURITY_AUDIT.AUDIT$JOURNAL (the _AUDIT is omitted in some latest versions of OpenVMS) The security auditing information is stored in this file. You should determine how long you wish to keep this data, and how often you start a new file. A new file is started with `SET AUDIT/SERVER=NEW`. [Reports are generated with `ANALYZE/AUDIT`.]

The system disk, cont'd.

SYS$MANAGER:ACCOUNTNG.DAT The system accounting records are kept in this file. You'll probably wish to treat this one similar to the previous security file. A new file is started with `SET ACCOUNTING/NEW`. [Reports are generated with `ACCOUNTING`.]

SYS$ERRORLOG:ERRLOG.SYS The error formatting process (ERRFMT) writes errors and timestamps to this file. To start a new one, simply rename or delete the old one; ERRFMT will create a new one if it doesn't exist. You'll probably wish to coordinate this with your field service engineer. [Reports are generated with SPEAR or `ANALYZE/ERROR`.]

Fragmentation

Problems with disk fragmentation

- Additional seek time – since the logical blocks of a file are scattered, simple sequential reads will be penalized
- Split I/O's – when you request an I/O that spans file extents, it must be split into an I/O for each extent
- Window turns – file extents are mapped in a window; when you have more than will fit, it causes a "window turn"
- Contiguous space requirements – some files and all directories must be contiguous, and excessive fragmentation will cause failures

Fragmentation, cont'd.

Checking fragmentation

- The "split transfer rate" from **`MONITOR IO`** gives a good indication of system-wide fragmentation, but the disks can't be determined
- The "window turn rate" from **`MONITOR FCP`** is the next best indicator, but is of no more help
- The performance analysis reports from the POLYCENTER Performance Solution (a.k.a. DECps, VPA) will give much more detailed information by disk as the fragmented files are used
- You can check the number of retrieval pointers a file has with: **`DUMP/HEADER/BLOCK=COUNT=0 filename`**
- Other software like FRAG, or the DEC File Optimizer

Balance I/O

Disk level The most obvious way to distribute I/O load is by placing the most active files on different disks. The hardest part is determining which files are most active, and application restrictions on placement. `MONITOR DISK/ITEM=ALL` is a good tool for measuring disk load. It gives both the I/O operation rate and I/O request queue length. The request queue length is how many I/O's are waiting on another to complete.

Controller level Another often overlooked way to balance I/O load is across controllers. Whether it's the requestor boards in an HSC, the CI controllers, DSSI or SCSI busses, spreading the load will help your high I/O applications.

Balance I/O, cont'd.

Bound volume sets With traditional VMS bound volume sets, you can distribute the load for one volume between disk drives. This doesn't work well for user drives as the most active files will tend to be on the same drive. However you can actually place an index file's areas on different drives, keeping the index on a smaller, faster drive than the data area for example.

Stripe sets A very good way to balance I/O is disk striping. It looks like one big disk, but the logical block numbers (LBN) are mapped across the drives in "chunks" (typically one track). This works well for traditional user or production packs with minimal management overhead.

Chunk = 3 Ex:

Physical	1	2	3	4	5	6	7...
1^{st} Disk	1	2	3	13	14	15	25...
2^{nd} Disk	4	5	6	16	17	18	28...
3^{rd} Disk	7	8	9	19	20	21	31...
4^{th} Disk	10	11	12	22	23	24	34...

RMS Indexed Files

“I don’t need to know about RMS indexed file tuning, I don’t use them.”

If you use OpenVMS, you use RMS indexed files.

- VMS uses a couple dozen of them
- Some are hit regularly
  - User authorization files (SYSUAF, RIGHTSLIST, NETPROXY)
  - Mail files (VMSMAIL_PROFILE.DATA, various MAIL.MAI)
  - Queue manager data files
  - Audit server and accounting files
- Don’t overlook RMS tuning on these system files

RMS Indexed Files, cont'd.

Why you need to tune RMS indexed files:

- By default, files are optimized for space, not performance. This is mainly a holdover from the '70's, when disk space was more precious than time.
- When records are added to indexed files, bucket splits occur that may increase the index depth, requiring more I/O's to locate a single record.
- When records are deleted, the locations are just marked deleted in the index; you may have a bucket with nothing but deleted index records! Using `CONVERT/RECLAIM` will return these totally empty buckets, but do nothing for others.

The ANALYZE, EDIT, and CONVERT cycle

Indexed files are reorganized with CONVERT

1. `ANALYZE/RMS/FDL SAMPLE.DAT`
   Produces an FDL file with analysis statistics.

2. `EDIT/FDL/ANALYSIS=SAMPLE.FDL/SCRIPT=OPTIMIZE SAMPLE.FDL`
   Invokes the FDL editor in interactive optimize mode. Alternately, you can use the following in batch mode:

   `EDIT/FDL/ANALYSIS=SAMPLE.FDL/NOINTERACTIVE SAMPLE.FDL`

3. `CONVERT/FDL=SAMPLE.FDL/NOSORT/STAT SAMPLE.DAT SAMPLE.DAT`
   Performs the actual file optimizations.

Index files and compression

There are three kinds of compression for indexed files:

Index compression is used to compress keys appearing in the index. It uses the last key and, taking advantage of the fact that the keys will be in alphabetical order and therefore somewhat similar, does front-end compression by removing characters that this key has in common with the one before. It also does compression of repeating leading or trailing characters.

Data key compression is used to compress the the key that is stored with the data record. It works exactly the same as index key compression.

Data record compression is used to compress the actual record (minus the key, which is actually stored first, though it is inserted in its proper place in the record before being returned to the user). It consists of compressing repeating characters.

The evils of index compression

Since what's stored for a key in a compressed index depends on the previous key in that bucket, the bucket must be searched sequentially for the key, expanding the keys as it goes. Without index compression, a binary search is used. So, unless you can get 200% compression or better, **just say "no" to index compression**.

Note that, after you've turned off compression, using CONVERT or even EDIT/FDL won't turn it back on. So, it's best to first do an interactive edit of the FDL to disable compression. It's also worth noting that a non-interactive EDIT/FDL will assume that no records are going to be added to the file. So, you might be better off just using CONVERT after you've optimized the file the first time, as it will keep areas allocated the same size.

RMS global buffers

Advantages:

- Multiple users share the same physical memory
- Index buckets are likely to already be buffered; lower disk I/O
- More buffers available; lower disk I/O
- If entire index is buffered, lowers I/O's to 1 per data bucket

Disadvantages:

- Increased locking activity
- Not cluster-wide; cache becomes invalid frequently when modified across nodes
- Depends on multiple accessors on same node for performance gain
- Requires global pages, global sections, and global page file space

RMS global buffers, cont'd.

- Enabled by **`SET FILE/GLOBAL_BUFFER=buffers`**
- Buffer size equals largest bucket size
- Ideally have whole index plus one data block for each simultaneous user
- Minimally need one per index level per key plus one
- Remember to change SYSGEN parameters based on global buffer requirements

RMS global buffer SYSGEN parameters

RMS_GBLBUFQUO The RMS global buffer quota limits the total number of RMS global buffers available system-wide at the same time. So, while you could set files for more global buffers than the quota, you could not have them all open at the same time.

GBLPAGFIL The global page file quota limits the number of writable global pages which can be written to the page file. It should be at least as big as RMS_GBLBUFQUO times the average buffer size in pages.

GBLSECTIONS The number of global sections available are used by all global memory entities, like installed images. Each open file that has global buffers enabled requires a global section. You should increase GBLSECTIONS by the number of files with global buffers that you expect to have open at the same time.

RMS global buffer SYSGEN parameters, cont'd.

GBLPAGES The number of global pages limits shared memory for all uses, such as installed images as well as RMS global buffers. It needs to be increased to the number of global pages required by the global buffers, like GBLPAGFIL. Two extra pages are required for each global section, to act as guard pages.

LOCKIDTBL The number of lock manager locks available. This value should be *increased* by the amount of RMS_GBLBUFQUO.

LOCKIDTBL_MAX Whenever you increase LOCKIDTBL, you should increase LOCKIDTBL_MAX by the same amount.

RESHASHTBL Whenever you change LOCKIDTBL, you should re-evaluate the resource hash table size. It needs to be one quarter of LOCKIDTBL, rounded up to the nearest power of two.

Monitoring RMS global buffer usage

- `INSTALL LIST/GLOBAL` will list all global sections. The ones named starting with RMS$ followed by a hexadecimal memory address (the FCB) are from RMS global buffers. Mapping the FCB (file control block) back to a disk and file name is not simple, however.

- `MONITOR RMS/ITEM=CACHING` on the files in question. A `SET FILE/STATISTICS` must have been done previous to opening the file. This is the best way to determine what effect the global buffers are having. By changing the number of buffers and comparing results, you should be able to find an optimal range. Note that each file set for statistics gathering requires yet another global section and global page while it's open, whether you're monitoring it or not.

Questions?

Questions?

Questions?