IMAGE Performance (part 2)

Ken Paul
Adager Corporation
Sun Valley, Idaho 83353-3000 • *USA*
http://www.adager.com

This month I continue the discussion I started two months ago concerning IMAGE performance. First I will review the main points I made in the first part of this article.

Adding or Retrieving?

This is the major question which must be answered when trying to determine an IMAGE performance problem. In the first part of this article I covered the possible reasons why adding entries to an IMAGE database may be a performance problem. The three reasons why trying to add entries to an IMAGE database may adversely affect performance are: a) the possibility of clusters with an Integer Key, b) a Sorted Path or c) a master dataset which is becoming too full.

Retrieving Detail Entries

If any of your applications takes a long time to retrieve entries from your IMAGE database, most likely the entries do not have a good locality for the access that your application requires. This means that a detail dataset is in great need of repacking its entries along the path accessed by the application. I discussed repacking and Adager's REPACK DATASET function in great detail several months ago. I will only synopsize here.

Most IMAGE database retrievals consist of obtaining an entry from a master dataset and then retrieving all of the detail entries that are related to this master entry. This is known as a "chained access". As chains within a detail dataset become longer the entries of the chain are usually spread farther and farther apart within the detail dataset. In the extreme case, this causes IMAGE to do an I/O for every entry of the chain. If we can somehow get all the entries of a given chain physically next

to each other, IMAGE would not have to issue as many I/Os to obtain all the entries on a given chain.

This is exactly what Adager's Repack Dataset function does. You specify the path (primary or not) along which you would like the entries repacked. But it is also important to maintain a periodic repacking schedule because performance will begin to deteriorate as new entries are added and deleted from the dataset.

Retrieving Master Entries.

People usually tend to blame master datasets for their retrieving performance problems but this is rarely the case. An analysis of your master datasets with DBLOADNG or DBLOADII from the CSL or HOWMESSY from Robelle will show you that your average synonym chain is rarely over 2. This means that at most IMAGE may incur two I/Os to retrieve a master entry but it is more likely to only take one I/O since IMAGE tries to place secondaries in the same block as their primary.

Beware of R4 keys.

The only time that I have seen a performance problem with retrieving entries from a master dataset was when an R4 key was used. Using R4 keys is 1000 times worse than using Integer keys and fortunately not many people use them. R4 keys are bad because IMAGE treats them as Integer Keys when it is trying to place an entry within the master dataset. Integer keys are not "hashed" but are "moduloed" and only the rightmost two words of the R4 are used. Since these two words do not change much from R4 key value to R4 key value all entries tend to end up with the same address. This obviously produces a very long synonym chain. R4 keys should be avoided.

HWMPut Flag

As of MPE/iX release 5.0 TurboIMAGE has a new control flag which is accessible through DBUTIL. The HWMPut flag allows control of where to place a new detail entry. When the flag is enabled, DBPUT will place the new detail entry at the HighWaterMark until it reaches the file limit. After this, DBPUT will start using the delete chain. Currently, DBPUT checks first if there is an available spot in the delete chain before placing an entry at the HighWaterMark. The purpose of this flag is to improve the data entry locality (by not recycling random locations whose entries have been previously deleted) and to improve chained access performance. For details on how to use the HMWPut flag, please consult the MPE/iX 5.0 Communicator.

You should not consider the HWMPut enhancement as a replacement for periodic repacking of your detail dataset. Before enabling your databases for HWMPut you should consider the amount of serial reads that your applications do on all the detail datasets to determine if the cost of continuously increasing the number of records that your serial reads perform (because all the deleted entries will not be recycled) does not offset the gains of not using the delete chain. In detail datasets enabled for dynamic expansion, you should also be careful that the size of the datasets does not automatically increase to unreasonable limits before you decide to repack them.

Dbloading and Robelle's HowMessy

In the first part of this article I mentioned the DBLOADNG from the CSL and HowMessy from Robelle (which we include in your Adager tape as a courtesy from Robelle) as possible tools to obtain a snapshot of some of your IMAGE database parameters. Both of these programs are available in both compatibility and native mode. The native mode version of DBLOADNG is called DBLOADII and it's also available from Interex.

Monitoring your IMAGE databases

Adager has a client/server option (which requires a PC running Microsoft's Windows and WRQ's Reflection for Windows version 4.0 or later) that allows you to continuously monitor and manage the performance-related parameters of your database. For instance, you can keep track of capacities, disc utilization, messiness of the detail dataset chains and amount of secondaries entries in master datasets. This option was described in detail in a previous Adager Column.

If you have IMAGE performance problems and would like to discuss your particular database, contact Adager technical support. To help us in the discussion, remember to observe whether your performance problem occurs primarily while *adding* entries or while *retrieving* entries.