HOW TO GET MORE FROM YOUR CORE MEMORY

PIERRE SENANT

P. SENANT
COGELOG
ETUDE ET RÉALISATION DE SYSTÈMES INFORMATIQUES
AVENUE DE LA BALTIQUE, Z.A. DE COURTABOEUF
91940 LES ULIS, FRANCE

COGELOG/ Etude et réalisation de systèmes informatiques
Avenue de la Baltique, Z.A. de Courtabœuf, 91940 LES ULIS / FRANCE
Tel.: (6) 907.70.79

HOW TO GET MORE FROM YOUR CORE MEMORY

OR

CFS/3000

A CORE RESIDENT FILE SYSTEM

Pierre Senant

A LITTLE HISTORY ...

A data processing system ordinarily uses three main types of memory, which have each a different speed level. Besides, the byte cost of these types of memory varies with their access speed. Here are the three types of memory in decreasing cost order :

1. Core memory. The latest technology uses integrated circuits and the access time is calculated in micro-seconds. The purpose of this memory is to contain program segments during the time they are executed, as well as a part of the data to be handled. The life time of these elements in memory is between a few milli-seconds, and many hours.

2. Auxiliary memory : Magnetic discs are normally used. The time to access information is calculated in milli-seconds. The purpose of this kind of memory is to save programs and data to be used during a given period, which can be from a few minutes to many months.

3. Archival memory : This can be done through many devices, but the most common is the magnetic tape. Since the access to data is performed serially, the access time to data can be from a few seconds to many minutes. This low-cost type of memory is used as back-up memory, and for saving all data currently unused.

When we consider the evolution of these different types of memory during the last past years, we observe a fall in prices for all types, but a particular drop for the core memory.

This drop has considerably changed the physionomy of data processing systems during the last years. The HP-3000 system remained in the swim. When it came out in 1972, the maximum memory size supported was 128 K bytes. The model of 1981 can support up to 4000 K bytes.

In addition, this trend will probably be confirmed in the next years, and an HP-3000 with 20.000 K memory will be common soon. The advantages of the increase of the core memory are evident :

. The amount of code that can be put in memory at a given time is larger. This implies a fall in data segment swapping, which dramatically reduces the disc overhead and increases the throughput of the system.

. The amount of data handled in one time can theoretically be increased, by using large file system buffers. However this technique gives disappointing results, especially in random access.

Most data processing applications are now using more and more interactive mode, instead of batch mode. Improving batches is important, but improving on-line programs is vital. Batches can run in off-peak hours, and most of the time the jobs can be done. But in interactive applications, each second of response time lost must be multiplied by the number of users, and employer's time is getting expensive.

So, the benefit of increasing core memory comes almost exclusively from improving the programs flow. Suppose we could increase indefinitely the core memory size, we would reach a critical point where adding a memory module would not affect the response time, because all program segments are already in core.

Another important factor bears heavily on the response time disc accesses due to transactions of application programs. An on-line program using a data base system (IMAGE 3000 for example) may be very greedy in disc accesses. This overhead

is independant of the number of program segments in memory and it becomes the actual bottleneck. This makes unprofitable an increase of core memory.

When a system has reached this level of evolution, one possible way to reduce the overhead is to suppress some disc accesses. At first sight, this load seems to be incompressible. If we assume the files and the data bases have been correctly organized, and application programs have been written with shrewdress , what can we do ?
However, when we examine all kinds of files involved in an application, we are surprised by their diversity, in the size, in the organization, and in the usage.
In fact, most of them are small enough, and are so frequently accessed that we would do better to make them core-resident.

NOW THE PRESENT : CFS/3000

The idea to make some files core-resident might not be very original, but it certainly remained a dream until to day.

Now the dream becomes reality with CFS/3000.
In my opinion, a real in-core memory file system must follow the following principles.

. It must be program independant.
  Allowing programming people to decide if a file must be core-resident or disc resident would be catastrophic.
. A core resident file must be accessible from all processes.
  The duplication of data is not acceptable.
. It must be reliable. Data integrity must not be affected, as well as the process independancy.
. It must respect all security and privacy provisions of the file system, as well as those of subsystems (IMAGE, KSAM).
. It must be fully controlled by the System Manager, who must know at any time.
  - the name and the sizes of core-resident files
  - the total memory size used
  - access frequency of any data-set.

In addition, the System Manager can decide to put in or out of core-memory any file at any time without disturbance in operation.

The conception of CFS/3000 has been founded upon these principles.

HOW TO USE CFS/3000

This product is intended to optimize both batch programs and on-line programs. In all cases, a good knowledge of application programs and system management will be required.

. In a batch environment, the execution time can be dramatically reduced. It will be strongly recommended to run one program at a time, in order to devote the maximum core-memory to data files.

. In an interactive environment, the tuning must be more accurate.

You will have to use the utility program IOSTAT2 to determine how busy your discs are, and what they are doing (swapping or accessing data files).
If the swapping is low, you may use CFS/3000. You will select the file to make core-resident by considering their impact on the response time.
With CFS/3000 you will be able to make trials without stopping the daily operation.

As a first step, you may declare core-resident some small files frequently accessed like.
- . Tables of parameters
- . Automatic master data sets (IMAGE)
- . KSAM key-files

In a second step, you will be able to design your future applications, by taking into account this new possibility.