THE 5 MINUTE RULE FOR TRADING MEMORY FOR DISC ACCESSES
and
THE 10 BYTE RULE FOR TRADING MEMORY FOR CPU TIME

Jim Gray
Franco Putzolu
Tandem Computers, Cupertino, CA, USA

ABSTRACT: If an item is accessed frequently enough, it should be main memory resident. For current technology, "frequently enough" means about every five minutes.

Along a similar vein, one can frequently trade memory space for cpu time. For example, bits can be packed in a byte at the expense of extra instructions to extract the bits. It makes economic sense to spend ten bytes of main memory to save one instruction per second.

These results depend on current price ratios of processors, memory and disc accesses These ratios are changing and hence the constants in the rules are changing.

THE FIVE MINUTE RULE

If data is disc resident it must be moved to main memory prior to being read or written. This movement costs both time and money.

In some situations, response time dictates that data be main memory resident because disc accesses introduce too much delay. These situations are rare. More commonly, keeping data main-memory resident is purely an economic issue. When does it make economic sense to make data resident in main memory? A good rule of thumb is:

> THE FIVE MINUTE RULE
> Data referenced every five minutes
> should be memory resident.

Twenty years ago the .5 second rule applied, twenty years hence it is likely to be the 5 hour rule. The exact constant is very dependent on current price ratios. To make the argument concrete, we will use Tandem 1986 prices here As discussed later, similar conclusions result from current IBM or DEC prices.

The derivation of the five minute rule goes as follows: A disc, and half a controller comfortably deliver 15 random accesses per second and are priced at about 15K$ [Tandem] So the price per disc access per second is about 1K$/a/s. The extra CPU and channel cost for supporting a disc is 1K$/a/s. So one disc access per second costs about 2K$/a/s.

A megabyte of main memory costs about 5K$, so a kilobyte costs 5$.

If making a 1Kb data record main-memory resident saves 1a/s, then it saves about 2K$ worth of disc accesses at a cost of 5$, a good deal. If it saves .1a/s then it saves about 200$, still a good deal. Continuing this, the break-even point is one access every 2000/5 ~ 400 seconds.

So, any 1KB record accessed more frequently than every 400 seconds should live in main memory 400 seconds is "about" 5 minutes, hence the name. the Five Minute Rule.

For smaller records, the break-even point is longer (1 hour for 100 byte records) and for larger records the break-even point is shorter (2 minutes for 4K records).

At a certain point the record size exceeds the disc transfer size. For example, page-faulting a 100K program requires twenty five 4K disc reads. So above the transfer size (4K in Tandem's case) one must use the rule for the transfer size (2 minutes in Tandem's case).

A more formal derivation and statement is:

Let:

- RI: expected interval in seconds between references to the data (second/access).
- M$: the cost of a byte of main memory ($/byte)
- A$: the cost of a disc access per second ($/access/second).
- B: the size of the data (byte).

Bmax: the maximum transfer size from disc (byte).

Then, assuming B < Bmax, the savings in dollars of keeping the record B main memory resident is.

$$\frac{A\$}{RI} - M\$*B$$

At the break-even point, this expression is zero. Solving for RI gives

$$RI = \frac{A\$}{M\$*B}$$

Substituting for the Tandem numbers (A$ = 2000, M$ = .005):

$$RI = \frac{400{,}000}{B} \text{ second/access}$$

As shown in Figure 1, the Five Minute Rule only approximates a particular region of the curve: B near 1K. Using the five minute rule for larger data items anticipates the advent of cheaper memory.

We used standard PRICEs. If we had used standard COSTS (about 10% of price) then the ratios would have been similar.

The Five Minute Rule also seems to apply to IBM systems (prices are uniformly higher for IBM 30XX machines and about the same for IBM 43xx machines) and for mini-computers (where everything is uniformly less expensive).

The Five Minute Rule does not apply to personal computers for two reasons. First one cannot add and subtract discs from PCs and workstations at will. Typically, one has the choice of zero or one hard discs. Second, memory and disc economics are different for PCs -- a disc costs about the same as a megabyte of main memory.

In the next decade 64Mbit memory chips are likely to reduce the memory vs disc-access cost ratio by another factor of 100. This will give rise to the 5 hour rule.

The following case study illustrates an application of the Five Minute Rule. A designer wanted to keep his entire database main memory resident. The database had 500,000 records and each record was about 1000 bytes long, so the whole database was about 500Mbytes. The following argument convinced him to adopt a hybrid disc-memory design.

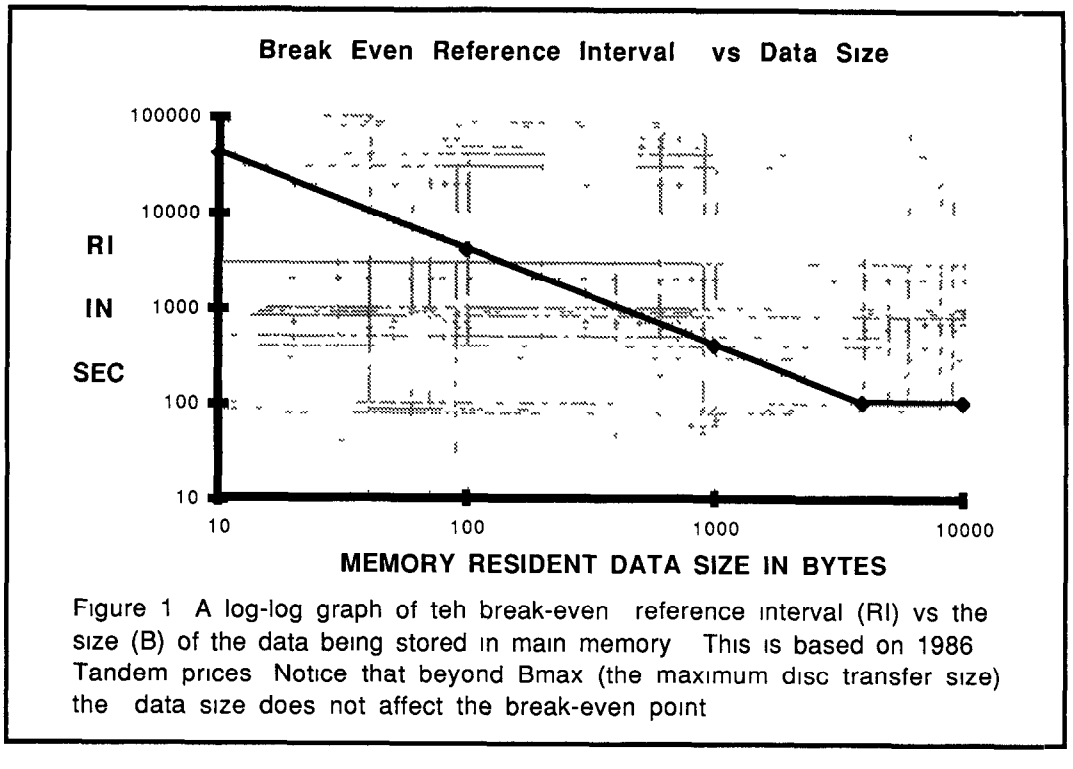

Figure 1 A log-log graph of teh break-even reference interval (RI) vs the size (B) of the data being stored in main memory This is based on 1986 Tandem prices Notice that beyond Bmax (the maximum disc transfer size) the data size does not affect the break-even point

The application transactions were quite simple Almost all the transactions accessed a single record and demanded one second average response time. The transaction used 40ms of cpu time and 30ms of disc time The total application had a 600 transaction per second peak load.

The all-in-main-memory design needed about 36 VLX processors, each with 16MB of main memory Two mirrored discs stored the entire database, its indices and the programs. The discs would be idle during normal operation since the database would be memory resident.

The five minute rule suggested that only records with a reference interval of 5 minutes or less should be in main memory. Based on estimates, later confirmed in another videotext application [Fromm & Hoelsken], very few records are frequently referenced. They observed that 3% of the records got 80% of the references. A statistical estimate indicated that at 600TPS, only 30,000 of the 500,000 records would be referenced twice in a five minute interval. In addition, this 6% of the data would get 96% of all accesses so only 24 disc

accesses per second were introduced (4% of 600TPS). Two disc drives storing the data could comfortably support this read rate. The consequent design saved 470Mb of main memory and six cpus. This was a 3.5M$ savings (a 30% savings) over the entirely main memory design.

Any application's database reference string can be used with this logic to compute the optimal size of disc cache and optimal number of disc arms.

This Five Minute Rule applies equally well to virtual memory management. If a virtual memory page (typically 4K bytes) is referenced every 2 minutes, it should stay memory resident. Hence a CLOCK virtual memory algorithm [Carr] should be given enough memory to cycle once every minute at peak loads or one should try to detect such "hot" pages (using a 2 minute history string) and treat such "hot" pages specially.

THE TEN BYTE RULE

Another interesting question is: "When does it make economic sense to use more memory to save some cpu power?", or conversely save some memory at the expense of some cpu cycles? This issue arises in code optimization where one can save some instructions by unwinding loops, and in data structure design where one can pack data at the expense of extra cpu instructions to mask and shift bytes to extract the bits.

The argument is similar to the Five Minute Rule. One picks a certain price for memory (say 5K$/MB) and a certain price per MIP (say 50K$/MIP). This means that 10 bytes cost about .05$. Similarly one instruction per second costs about .05$. So 10 bytes costs about the same as 1 instruction per second. This gives rise to the ten byte rule:

> THE TEN BYTE RULE
> Spend 10 bytes of main memory
> to save 1 instruction per second.

The Ten Byte Rule is applied as follows. Let

- I: how many instructions are saved by the new design (instructions)
- F: how frequently the instruction sequence is executed (1/second)
- S: how many bytes are saved by the new design (bytes).

The product of I and F is the instruction savings of the change. It will be negative if the design adds instructions.

Using the Ten Byte Rule compare S (the memory savings) with the instruction savings by dividing S by 10. The net savings is:

```
I * F - S / 10
```

If it is a large positive number, then the new design provides a large savings.

As an example, suppose the dispatcher of an operating system has a code sequence like:

```
LOAD        BYTE
MASK        FLAG
BRANCH ON   NONZERO
```

Suppose the dispatcher is invoked 1000 times each second.

* If flag were stored as a byte it would avoid the mask step and hence save 1000 instructions per second.

* This translates to about 10000 bytes of storage based on the Ten Byte Rule.

* If flag were stored as a byte it would use eight times the storage. If there are 100 processes in the processor, this translates to about 88 extra bytes.

* Since we have a 10,000 byte budget, this is a profit of 9912 bytes for an 88 byte investment. A good trade -- about a 100:1 return on investment.

On non-RISC machines, the MASK instruction may use 2 micro-clocks while the average instruction uses 6 micro-clocks. In this case, one needs to weight the saved instructions with their micro-clock cost. That is, in the example above, would save only 2/6 of an

instruction each time we saved a MASK step. So the "real" savings on the hypothetical non-RISC machine would be only (100*(2/6)):1 ~ 33:1. Still a good deal.

THE FUTURE

The idea underlying these rules is a way of evaluating the economic tradeoffs of design decisions. As technology changes the price-performance ratios, these design rules must also change.

Over the last two decades, disc module capacity and memory chip capacity have grown about a factor of 100 every decade with a proportional drop in price per megabyte. Processor speeds have grown a factor of ten every decade with a consequent drop in price per mip. Disc accesses have had a relatively constant cost over this period, discs continue to deliver about 15 accesses/arm/second random and 60 accesses/arm/second sequential.

Extrapolating these trends to 1996, the 5 minute rule will become the 5 hour rule -- data used once every five hours will be main memory resident. In that time-frame, there will be considerable interest in optimizing cpu cache occupancy since main memory will begin to look like secondary storage to processors and their memory caches.

Similarly, the 10 byte rule is likely to become the 100 byte rule -- one will be willing to squander main memory in order to save a few instructions.

ACKNOWLEDGMENTS

Dina Bitton, Haran Boral, Fritz Graf, and Pete Homan pointed out errors and suggested many improvements to earlier versions of this paper.

REFERENCES

[Carr] Virtual Memory Management University Microfilm, 1984.

[Fromm & Hoelsken] Fromm, H , Hoelsken, H., "Beilschirmtext Usage Characteristics and Performance Aspects of the German Videotext System" Digest of Papers, Compcon 87, IEEE Computer Society Press, order #764, pp. 152-160, Feb. 1987.

[Tandem] Tandem Product and Price Guide, Tandem Computers Inc. Cupertino, CA. Sept 1986.