A Comparative Study Regarding a Memory Hierarchy with the CDLR SPEC 2000 Simulator

O. Novac [*], M. Vlăduțiu [**], St. Vari Kakas [*], Mihaela Novac[***] and M. Gordan [***]
[*] Department of Computers, University of Oradea,
[**] Department of Computers, "Politehnica" University of Timişoara,
[***] Department of Electrotechnics, University of Oradea

***Abstract*-We have built a simulator named, CDLR SPEC 2000. This simulator is based on traces for systems with cache memories. With CDLR SPEC 2000 program we can study, for a memory hierarcy, the next parameters: maping function, block size, writing strategies, replacement algoritm, size of cache memory, number of cache sets (for the set associative caches), number of words form a block. The simulator can be used for the study of cache memory behaviour. Also the CDLR SPEC 2000 program, introduce the calculus of CDLR of a memory hierarchy.**

A common metric used for the assessment of overall reliability, in a memory hierarchy is the Mean Time To Failure (MTTF), but it doesn't take into account for the time of data storage in each level. We propose another metric, Cache Data Loss Rate (CDLR), for this reason. We will derive a recurrent formula for computing CDLR, and we will validate it by computer simulation.

I. Introduction

All high performance microprocessors use a hierarchy of cache memories to hide the slow access to the main memory [1]. Hence the memory hierarchy is important part in a computer system With each new generation of integrated circuit technology, feature sizes shrink, creating room for more functionality on one chip. We see a fast growth in the amount of cache that designers integrate into a microprocessor to gain higher performance. Hence, caches occupy a substantial area of a microprocessor chip and contain a high percentage of the total number of transistors in the chip. Consequently, the reliability of the cache has a big impact on the overall chip reliability [5]. Practically if a computer system has a fast and efficient processor, if the memory hierarchy is slow and inefficient, memory access bottlenecks will arise and the overall system performance will be low. For this reason in any system design an adequate attention should be paid to the design of memory hierarchy subsystems, such as memory interface, cache, paging mechanism, TLB, CPU memory hierarchy support registers.

Caches are used to bridge the speed gap between microprocessors and main memory. Without them, the processor can still operate correctly but at a substantially lower performance. Most of the architectures are designed such that the processor can operate without a cache under certain circumstances. Therefore, the obvious solution to a faulty cache is to totally disable it. In set-associative caches, a less extreme solution is to disable one unit (way) of the cache [4].

The memory devices of a computer system are organized in a hierarchy in order to achieve a good rate between performance and cost per bit. Proceeding down in the memory hierarchy, its reliability improves owing to the storage technology in different levels. There is of course a tradeoff between high reliability and performance, which is influenced beside the construction, by the transfer policy used among levels. Transfer policy is important because it directly affects the reliability of the overall hierarchy. The first possibility is to write through to the most reliable, lowest level every time a data is transmitted from the CPU. This policy offers good reliability, but bad performance (high overhead for transferring data). The second possibility is to write back data to lower level only when needed (for instance based on the amount of data in a level). This method yields a more reduced reliability (as data stays longer in a less reliable level), but better performance (less overhead for transferring data). The third possibility is the delayed write, when data is written from level L to level L+1 after a time denoted $delay_L$. So $delay_L$ is the age of the data before it leaves level L and is written to level L+1. We can observe that $delay_{L,}$ monotonically increase with L.

II. MTTDL And CDLR Metrics

MTTDL (Mean Time To Data Los) is actually the MTTF (Mean Time To Failure), the mean time to a hierarchy breakdown. If we suppose a high serial reliability of the system's block diagram and for each level we have an exponential distribution for a given MTTF, the MTTDL for a hierarchy with N levels is given as in [2]:

T. Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, 369–372.

$$MTTDL = \frac{1}{\sum_{L=1}^{N} \frac{1}{MTTF_L}} \quad (1)$$

In computing the MTTDL we can observe that it is limited by the reliability of the least reliable level. One can say that MTTDL is relevant only when the data loss is proportional. The MTTDL is general and it can not distinguish between the loss of a great amount of data (due to a disk failure) and a data loss due to a memory error.

For this reason we will propose a new measure - CDLR (Cache Data Loss Rate) which represents the data loss ratio during the memory hierarchy failure. The mean access time for a reliable architecture with N levels is calculated as the arithmetical mean of access times:

$$\sum_{L=1}^{N} hitRate_L x Acces_L \quad (2)$$

The DLR (Data Loss Rate), for a reliable memory hierarchy is the weighted sum of the data loss duration when a level fails and is given by the following formula (3) [2]:

$$DLR = \sum_{L=1}^{n} \frac{duration_L}{MTTF_L} \quad (3)$$

We can say that $duration_L$ implies a special case, that there is no lower level in which data can be transferred.

A first method to tackle this problem, is to suppose that the lowest level has an infinite MTTF and using a stocking technology that closes to this ideal (relatively to the reliability of the other levels).

Another approach is that of attributing to $duration_L$ the quantity of useful data that might be lost in case of a level failure. For example perhaps only the last year's data might be useful in the case the magnetic tape.

Both approaches lead to the following relation:

$$\sum_{L=1}^{N} \frac{f_L}{MTTF_L} \quad (4)$$

where f_L represent the amount of data lost in case level L fails. We can consider for example that f_L is a linear function that depends to $length_L$, such that in this formula newer data have a larger proportion than the old data.

The previous formula does not count the collateral malfunctions. If a malfunction affects several levels (such a malfunction is flooding), it must be modified in order to count a type a malfunction once. Thus a single malfunction would drop the MTTF on several levels and will affect the $MTTD_L$ and DLR several times. The way of correction the correlated malfunctions are to calculate the MTTDL and the DLR values, by adding the types of malfunctions rather than the levels of reliability:

$$MTTDL = \frac{1}{\sum_{altypeofmalfunctionsF} \frac{1}{MTTF_F}} \quad (5)$$

$$DLR = \sum_{altypeofmalfunctionsF} \frac{malfunction_F}{MTTF_F} \quad (6)$$

where $MTTF_F$ is the average time till the apparition of an F type malfunction and $malfunction_F$ is the duration of data loss, when malfunction F appear.

We shall use other formula than the one used in [2], to compute CDLR. For this purpose we shall use the following principle [3]:

No. losses in L = No. losses due to L + No. losses Due in (L+1) [No. losses in L induced from a loss in (L+1) – No. losses already produced in L from those induced by (L+1)

Starting from this principle, we obtain a recurring formula that has a better accuracy than the one used in [2]:

$$CDLR_L = \frac{delay_L}{MTTF_L} + CDLR_{L+1}\left(1 - \frac{delay_L}{MTTF_L}\right) \quad (7)$$

where $CDLR_L$, is a new term and represents the rate of data lost in level 1. (this formula does not allow us to take into consideration errors, data what appear in a data field already corrupted on a previous level). As I mentioned, the last level is a special case, because it has no lower level to transmit data. For this reason, we ought to take into consideration $MTTF_N$ is infinite, which means $CDLR_N = 0$ (otherwise the CDLR of the entire hierarchy would be of 100% and thus all the data would be lost in time). The evaluation of performance and reliability of some hierarchy of memory can be made now, by using the two sizes.

The CDLR SPEC 2000 program is built as the SMP Cache 2.0 program [6] for working with Traces, the difference is the fact that the CDLR SPEC 2000 program use traces of SPEC 2000 benchmark. The parameters of the CDLR SPEC 2000 program can be modified, by the user.

III. Simulation Example

For working with the simulator, it is necessary to use data files with the "calls" and memory addresses demanded by the processors during the execution of a program: the named memory traces. The trace files will allow us to emulate the programs with our CDLR SPEC 2000 program.

In case of a simulation with the CDLR SPEC 2000 program we must specify the context of simulation and the parameters for the simulation. (mapping function, block size, writing strategy and the replacement algorithm), in a input file. We will specify the mapping function: direct, set-associative or