Copyright Notice

On a warning from E.A.Hauck.

During my visit to Mission Viejo, last April, Erv Hauck made the passing remark that he did not believe that error recovery could compensate effectively for the ill effects of a basically unreliable storage technique. Intuitively I was perfectly willing to share that belief; this note reports on my efforts to justify it and to find the arguments that would change it into my considered opinion.

In the following I consider words of a length of n stored bits; with p0, p1, p2, etc. I shall denote the probability of no error, a one-bit error, a two-bit error, etc. If bit-errors are independent events occurring for each bit with a probability p --we shall call this "Assumption A"-- we have

$$p0 = (1 - p)^n \quad , \quad p1 = np(1 - p)^{n-1} \quad , \quad p2 = n(n - 1)p^2(1 - p)^{n-2}/2, \quad \text{etc.}$$

for large n and small p reasonably approximated by

$$p0 = 1 - p1, \quad p1 = np, \quad p2 = p1^2/2, \quad p3 = p1^3/6, \text{ etc.}$$

System 1, without rejected configurations.

To start with we consider a code that only corrects one-bit errors. (Such codes exist, e.g. for $n = 3$: "zero"= 000 and "one"= 111; then 001, 010, and 100 will be interpreted as "zero", and 110, 101, and 011 will be interpreted as "one".) With a memory with a microsecond cycle time and $p1 = 10^{-6}$, a one-bit error will be successfully corrected once every second, and under Assumption A an undetected error will occur once every 2,000,000 sec = 23 days. This may seem OK for the optimist, but it is not, on account of the absence of rejected configurations; suppose that --as a result of a drifting powersupply, say-- it gets worse and we go up to $p1 = 10^{-5}$: a one bit error will be corrected every 100 msec, an undetected error occurs every 20.000 sec = 5 hours, 30 minutes; when $p1 = 10^{-3}$, an undetected error will occur every 2 seconds! The absence of rejected configurations means that we are not warned for this deterioration and the resulting memory is something one cannot rely upon.

System 2, with rejected configurations.

We now consider a code that corrects one-bit errors, and detects two-bit errors. (Also such codes exist, e.g. for $n = 4$: "zero"= 0000 and "one"=

1111; any configuration with two ones and two zeroes will be rejected, such as 0110.) With the same microsecond cycle time and $p1 = 10^{-6}$, we have a one-bit error successfully corrected every second, under Assumption A a detected error every 23 days, and an undetected error once every 200,000 years. That seems safe, as a slowly increasing value of p, due to some technical degradation, may be expected to give the alarm of a two-bit error long before an undetected error has occurred. But it is, alas, absolutely unsafe, because in many --and in a sense: in all-- technologies, Assumption A is not justified: the storing and reading of n bits are not technically independent. We therefore consider for the sake of simplicity the other extreme --Assumption B-- "with a probability p the reading of a word will deliver n random bits".

Exploring Assumption B.

System 1 could have been improved by counting the number of corrections: under Assumption A a correction once every second would imply that the memory is not in too bad a condition (at least, if we think an error every 23 days acceptable --I don't actually, but that is now beside the point). Under Assumption B (because a random sequence is nearly sure to be interpreted as a one-bit error) the machine will perform a one-bit correction once every second, but whenever it does so, it is an erroneous correction: de facto the memory can be expected to make a fatal error once every second.

In order to estimate how System 2 would perform under Assumption B we must estimate how large the probability is, that a random sequence will be rejected. If each two-bit error is to be detected, any two correct codes must differ in at least 4 bit positions. For $n = 2^m$, the exact solutions are known: there are then 2^{n-m-1} different codes. As each code has 2^m+1 acceptable representations (the $n=2^m$ representations formed by changing one bit + the original code), the number of acceptable representations is $2^{n-m-1}(2^m+1) = 2^{(n-1)}(1+2^{-m})$, i.e. slightly more than half of the 2^n possible bit sequences. As a consequence slightly less than half of them will be rejected.

From this we must conclude that --regardless of the value of p -- when we start the machine, in 50 percent of the cases an undetected memory error has occurred before a memory error is detected: I cannot regard this as attractive either! (We could live with it if p is very small, i.e.

a highly reliable memory, but that was not the case we were considering!)

Assumption B --all bits random-- is, of course, a severe form of malfunctioning. But we don't get any solace from that: instead of random values for $n=2^m$ bits, we arrive at the same probability for rejection when choosing only $m+1$ bits randomly, and accepting the remaining $n-m-1$ bits as read from memory.

The moral of the story is, that Hauck's warning is not to be ignored!

* * *

The reason that my attention returned to Hauck's warning and that I tried to find its justification, was that I was (re)considering the relative merits of neutral, local redundancy --such as parity checks and their embellishments-- versus tailored, global redundancy, when our aim is to reduce drastically the probability that a wrong result will be mistaken for a correct one. Local error correction is in this respect harmful as soon as errors graver than those the detection mechanism can cope with, can occur as well. As the correction mechanism for single bit errors has enlarged the collection of acceptable representations, the probability that the computation proceeds with erroneous values increases with the length of the computation. But that is another story.

Burroughs
Plataanstraat 5
NL-4565 NUENEN
The Netherlands

prof.dr.Edsger W.Dijkstra
Burroughs Research Fellow