Correcting Detectable Uncorrectable Errors in Memory

Grzegorz Pawelczak, Simon McIntosh-Smith
University of Bristol

Introduction

With the expected decrease in Mean Time Between Failures (MTBF), Fault Tolerance (FT) has been identified as one of the major challenges for exascale computing. One source of faults is soft errors caused by cosmic rays, which can cause bit corruptions to the data held in memory. Current solutions for protection against these errors include Error Correcting Codes (ECC), which can detect and/or correct these errors. When an error that can be detected but not corrected occurs, a Detectable Uncorrectable Error (DUE) results, and unless checkpoint-restart is used, the system will usually fail. In our work we present a probabilistic method of correcting DUEs which occur in the part of the memory where the program instructions are stored. We devise a correction technique for DUEs for the ARM A64 instruction set which combines extended Hamming code with Cyclic Redundancy Check (CRC) code to provide near 100% Successful Correction Rate (SCR) of DUEs.

Previous Work

In their previous work, Gottsch et al. proposed Software-Defined Error-Correcting Codes (S-WD-ECC) which are capable of correcting DUEs by leveraging the properties of the underlying ECC and the side information about the data stored [1].

▶ Research focused on the 32-bit Single Error Correction and Double Error Detection (SECDED) code where for every 32-bits of data, an additional 7-bits of redundancy are stored to aid with error detection and correction.
▶ In the SECDED code an occurrence of a DUE means that there are multiple candidate codewords which are equidistant to the correct codeword.
▶ Since all the candidate codewords are equally likely, it is impossible to determine which is the correct one.
▶ By assuming that the data protected is MIPS program instructions, the candidate codewords which do not represent a valid instruction can be discarded.
▶ This approach reduced the search space to an average of 12 valid candidate instructions and along with an observation that some instructions are more likely than others, they were able to correct 34% of DUEs.

New Heuristic Techniques

We extend this work by investigating the performance on the ARM A64 Instruction Set which similarly to MIPS uses 32-bit instructions.

We introduce new heuristic techniques to reduce the number of candidate instructions in order to improve the SCR:

1. An addition of an Error Detecting Code (EDC) in order to reduce the number of candidate valid codewords.
   ▶ CRC32C is used as the EDC, where every N SECDED codewords are covered by a single CRC32C checksum.
   ▶ When a DUE occurs all valid SECDED codewords have to also produce a valid CRC32C checksum.
   ▶ CRC32C is used because it provides a high Hamming Distance (HD) [2] and modern architectures, such as x86 or ARMv8a, often provide hardware support for the computation of the checksum via intrinsics.
2. Filtering out codewords which are valid instructions, but are not deemed logical instructions.
   ▶ For example a branch instruction to an invalid address is an illegal instruction and should not be considered as a likely candidate.

Method

The following heuristic approach is used to assess an instruction:

1. SECCED Decode
   ▶ No error or detectable correctable error?
   ▶ Check that the valid instructions represent legal instructions
   ▶ Only one legal instruction?
   ▶ Only one valid instruction?
   ▶ Use VIXL* to convert candidate SECDED codewords into ARM A64 instructions
   ▶ Only one valid instruction?

2. CRC32C
   ▶ 7.96
   ▶ 7.16
   ▶ 1.00
   ▶ 1.00

Results & Conclusions

To evaluate the performance of these techniques, the binary programs from the /bin/ directory for the openSUSE Linux distribution are used as the training programs in order to find the most frequent instructions, while mini-apps from the UK Mini-App Consortium (https://uk-mac.github.io/) are used as the test programs. The DUE faults are injected by randomly selecting an instruction from the .text section of the test program and randomly flipping two unique bits in the instruction SECDED codeword.

* VIXL is a Runtime Code Generation Library by ARM which provides a disassembler (https://github.com/armvixl/vixl).

References


https://uob-hpc.github.io/