Amy Bui
Paper Review

1 Summary

Cores that don't count (2021): The authors give an introductory discussion on "silent" corrupt execution errors (CEE), errors that can appear suddenly and unpredictably within specific execution units in a core, also called "mercurial" cores. These kinds of defective cores are usually obscured by more obvious faults in computation, like software bugs, but are more visible now due to improvements in software development, the growing scale of industry server fleets, and "ever-smaller feature sizes that are pushing closer to the limits of CMOS scaling and with ever-increasing complex architectural designs". They view CEEs as a new cause of silent data corruptions (SDC), data corrupted during writing, reading, or resting that's not immediately detected and thought to happen randomly from alpha particles and cosmic rays. Since there is little documentation on corruptions like CEEs, there is no standardized or systematic methodology to test, detect, and isolate mercurial cores or make them CEE-tolerant. The authors characterize CEEs and point out significant challenges that make addressing them difficult with currently available tools. So this broad paper is the authors' "call-to-action" on a potentially growing issue in system designs.

2 Strengths

- Section 8 on related works touches on a variety of system failures and corruptions observed in the past and more recently, and the authors make good points on potential applicability (or lack thereof) of these errors and techniques other researchers used with regards to addressing CEEs.

3 Weaknesses

- There is no quantified data or any meaningful figures in their paper. Figure 1 only represents normalized CEE rates from "[their] fleet" with no information on specifications or operating conditions. The inclusion of data representing corruption trends, the scale of CEE rates, or even projected impact would help bolster their stance on the significance of CEEs.
- The authors reference a "corpus of code serving as test cases" that they used, but did not characterized use cases or what these test cases covered.

4 Rating: 3

5 Comments

The authors often allude to problems they observed in analyzing defects in their large server fleets and referencing other works relating to SDCs, applicability to CEEs, and the only other paper that discusses similar findings to theirs (Dixit *et al* on SDCs at Facebook/Meta). This is likely due to the constraints on what they were allowed to publish in this paper, where even "for business reasons, [they] are unable to reveal exact CEE rates". With a lack of data available, it makes it difficult to recognize CEEs as a non-negligible problem and an interesting, accessible avenue of research. However, the lack of specificity may mean that the authors anticipate working on a more detailed follow-up paper on the topic of CEEs with more of their findings. The authors already do a lot of work trying to characterize different problems (symptoms) of CEEs and the difficulties that come with root-causing CEEs. It is still clear though that the various, potential ways to detect, isolate, or mitigate CEEs are not very well-defined in the way they fleetingly touch on techniques or solutions that may or may not work to scale; this itself reveals the significance of researching and standardizing practices of addressing CEEs, rather than simply relying on dubious reports and miscellaneous tools and libraries. The paper's goals are better served by including things like corruption trends, the impact of utilized techniques, or simply any of the metrics discussed in Section 4, rather than reading as very anecdotal.

6 Notes

- The paper provides:
  - Context of the "silent" corrupt execution errors (CEE)
  - scale of these CEEs.
  - risks due to CEE challenges
  - suggestions to how to rapidly detect and quarantine mercurial cores, and how to create more resiliance to tolerate CEE.
- Processors are normally considered fail-stops. Sophisticated manufacturing tests (validation methods) detect defects, and any defects that escape or manifest with aging will trigger machine checks or give wrong answers to many kinds of instructions.
- Silent failures (silent data corruptions (SDC), silent **corrupt execution errors** (CEE)) are usually obscured by software bugs; but now mercurial cores are being seen because of a variety of factors:
  - ever-smaller feature sizes that push closer to limits of CMOS (Complementary metal-oxide-semiconductor) scaling, coupled with the every increasing complexity in architectural designs.
  - larger server fleets
  - increased attention to overall reliability
  - reduction in rate of software bugs due to improvements in software development

- CEE problem gets harder as the limits of silicon is pushed. They can no longer be ignore as unreliable hardware increasingly fail silently rather than fail-stop.
- These vulnerable chips need to be addressed in a way that doesn't become costly with replacing these chips or waiting for more resilient hardware to release.
- A high-rate of CEEs is considered by the authors to be another cause of SDC (i.e. data storage corrupted during reading/writing/rest without being immediately being detected). SDCs were previously thought to be caused by random things like alpha particles and cosmic rays and intentional practices like overclocking.
- **Sec. 2: Impact (symptoms) of Mercurial Cores:**
  - Symptoms:
    - Wrong answers, detected near immediately through self-checking, exceptions, seg faults. Can allow automated retries.
    - machine checks (more disruptive)
    - wrong answers, detected too late to retry computation
    - wrong answers, never detected
  - Wrong answers not detected right away can propagate.
  - Inferring root cause of mercurial cores is hard due to limited knowledge of underlying hardware. And wrong computation is not always detected. "CEEs are harder to root-cause than software bugs, which can be reproduced and debugged on another machine."
  - Identify mercurial core is usually just: code miscomputed/crash on xyz core; we can control which code runs on which cores and partial control of operating conditions (frequency, voltage, temp)
  - Failures mostly appear non-deterministically and at variable rates.
  - Faulty cores typically fail repeatedly and intermittently and get worse with time (also aging plays a role). In multicore sys, usually 1 core fails consistently.
  - Corruptin rate varies, depending on workload, f, V, and Temp. Difficult to tell if data patterns is contributing to corruption
- **Sec. 3: Are Mercurial Cores a New problem:**
  - storage and network problems are easier than computational errors because there is "right results" and simpler checks. CEEs require: automatic correction is triple work, and ost computational failures cannot be addressed with code-based approaches. S&N better tolerate low-level errors because they usually work n large chunks of data (disk blocks, network packets), so corruption check costs are amortized, which is harder to do at a per-instr scale.
  -
- **Sec. 4: The right metrics:**

  - potential metrics for CEE and challenges:
    * Fraction of cores/machines that exhibit CEEs. Challenges: Depends on test coverage, how many cycles devoted to testing, and ongoing arrival of new CPU parts to population.
    * Age until onset: Challenge: how long you can wait to actually start seeing CEE (frequent replacements/upgrades?), and requires continual screening over machine's lifetime.
    * rate and nature of application-visible corruptions (how often does CEE result of real world workloads?), and if corruptions are "sticky"/propagate through subsequent computations to create multiple application errors. Challenge: more of a property of applications than of CEEs.
  - quantifying metrics as they relate to CEEs is difficult and expensive (requires running tests on many machines for VERY long time for high-confidence results without even knowing what that time is, needs a concise set of tests that represent the complexity of real applications but we have a poor understanding of what triggers CEEs so we don't know how to create a small set of tests that will reliably measure these rates)
  - inaccuracies of measurements vs. cost of measurements.
  - risk assessment of CEE for a large fleet with various CPU types and various ages.

- **Sec. 5: Causes of Mercurial cores:**
  - come mercurial core CEE rates vary with what they are impacted by (f, T, V).
  -

- **Sec. 6: Detecting Mercurial cores:**
  - detecting mercurial cores mean we can isolate them and prevent further damage and to support deeper analysis.
  - Mercurial-core detection is challenging because it inherently involves a tradeoff between false negatives or delayed positives (leading to failures and data corruption), false positives (leading to wasted cores that are inappropriately isolated), and the non-trivial costs of the detection processes themselves.
  - Detection Processes:
    1. **automated vs. human screening**: automation in large fleet is ideal, but complexity-related causes of MCs suggest occassional new manifestations of CEEs,
    2. **pre-deployment vs post-deployment screening:** chip manufacturers have a lot of auto testing before chips ship, but it could be better because manufactuers don't have easy access to diverse large-scale workloads to learn about shortcomings in their tests, so cmanufacturer testing doesn't reflect real-world or they are not broad enough despite being very comprehensive already. Also, cores can become defective after considerable time has passed,

and new tests can be developed in response to newly discovered defects after deployment.

3. **offline vs. online screening:** online screening is free (except for power cost) but cannot always provide complete coverage of al lcores or all symptoms. Offline screening can involve exposing CPU to operating conditions of f, V, and T outside normal rande but it can be expensive.
4. **infastructure-level vs. application -level screening:** CEE detection can be done by infra (OS and daemon processes) or by some applications themselves (i.e. online). Infra can be more pervasive and dtect bugs in privileged execution, but application can be more focused and more easily fine-tuned and can enable app-level mitigations.

- existing sched mechanisms can remove a machine from a resource pool, but isolatingn specific cores is more challenging because it undermines a scheduler assumption that all machines of a specific type have identical resources.
- Hypothsized that a specific set of tasks that can run safely on defective core can be identified. but its not clear if this is reliable.

- **Sec. 7: Mitigating (tolerating) CEEs**
  - check correctness at application endpoints rather than in lower-level infra
  - system support for checkpointing to recover from failed computation by restarting on different core.
  - cost-effective application specific detection methods, to decide whether or not to continue past a checkpoint or retry
  - run computation on 2+ cores and see if they disagree (redundancy or results with most yields is most reliable)
  - These methods all have resource cost for duplicate computation and/or storage.
  - Hardware mitigations add costs but are more efficient than repllicating computations in software.

-