Compiler-directed Selective Register Checkpointing

Chengmo Yang
University of Delaware
chengmo@udel.edu

Abstract

As device feature sizes scale towards nanoscale, future computer systems are destined to suffer from various types of device failures that may occur during execution. Along with the projected high fault rate, we also expect a significant variance in fault duration. In addition to *transient* faults, typically caused by particle strikes, and *permanent* faults, that occur repeatedly after a device sustains irreversible damage, *intermittent* faults may occur frequently and irregularly for a period of time, commonly due to process variation or in-progress wear-out, combined with voltage and temperature fluctuations.

These reliability challenges are even more severe for embedded systems whose computation is confined to a limited or unstable power supply. For instance, cyber physical systems need to collect and process information using self-powered devices, such as wireless sensor nodes. Although sometimes these devices may be able to derive energy from external sources such as solar or wind, they suffer from power failures due to the instability of these power sources. As the computation state is not maintained when the power is off, not only may critical data get lost, but a fair amount of energy will also be wasted to restart the execution from the very beginning.

In the past, the tradeoffs between system size and complexity have dictated that the cheaper option is creating a compact, non-robust implementation and then replacing it when it fails. However, in more and more cases, it is either too expensive to access the system, or too expensive to remotely diagnose its failure state and repair it. Therefore, it is more and more preferable to develop designs that consume non-traditionally large areas, but are highly resilient to internal failures.

In this talk, we will present a technique capable of identifying a minimum set of register values for checkpointing. This technique exploits a widely-recognized program characteristic, that is, 90% of the execution time is spent on loops that constitute only 10% of the code size. Through analyzing the control and data flow of the hottest loops of a given application, this technique only selects a minimum set of registers for checkpointing. Specifically, register values are verified and checkpointed only if they impact program final results, or they are needed to recover the computation state. In this way, checkpointing overhead is minimized, the probability of *false errors* (i.e., faults that not impact program results) is maximally reduced, while high fault coverage is still guaranteed.

The proposed technique can be used in several scenarios. First, redundant execution is a traditional fault tolerant technique that delivers high fault coverage across the entire system against arbitrary faults. However, their applicability to embedded systems is quite limited due to their high overhead not only in verifying all instruction results, but also in unnecessarily recovering faults that will never influence subsequent execution. In contrast to the approach, the proposed technique selects not a *sufficient* set but a *necessary* set of registers for checkpointing. We observed that more than 75% of comparison and checkpointing overhead can be saved. This advantage will broaden the applicability of redundant execution to embedded systems of tight power and resource constraints.

Another scenario for the adoption of the proposed technique is to systems with non-volatile memories (NVM). NVMs are employed to overcome the energy and reliability challenges of embedded systems. Non-volatile devices, such as Phase Change Memory (PCM), Spin-Transfer Torque Random Access Memory (STT-RAM), and Ferroelectric Random Access Memory (FeRAM) offer a set of advantages, such as negligible standby power, high resilience to soft errors, as well as the ability to maintain states across interrupted executions. However, these devices are also constrained by limited write endurance, high write energy and slow write performance, which are extremely difficult to overcome when NVMs are deployed at the top level of the conventional memory hierarchy, that is, used as register files. As registers are updated extremely frequently (almost once per instruction), it is necessary to minimize the number of register write operations, which could be achieved with the proposed selective checkpointing technique. In this way, the slow write performance and limited write endurance of NVMs can be mitigated, thus qualifying them for various embedded systems.

CONF 'yy, Month d–d, 20yy, City, ST, Country.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn