Exploring the Effect of Compiler Optimizations on the Reliability of HPC Applications

Achievement: Demonstrated that compiler optimizations have an impact on the vulnerability of HPC applications in addition to performance, and therefore the need to consider tradeoffs between performance and vulnerability for critical application components.

Significance and Impact: The aim of this work is to quantify the resiliency of HPC applications to different compiler optimizations using accelerated fault-injection experiments.

Research Details:

- Utilized a compiler-level fault injection tool to analyze the effect of soft errors on parallel distributed HPC applications compiled using optimizations levels ranging from O0 to O3.
- Identified software attributes such as number of memory stores per instruction which are distinguishable with the use of different compiler optimizations and can affect application vulnerability.

Sponsor/Facility: Work was performed partially at Oak Ridge National Laboratory (ORNL) and was sponsored in part by the ORNL Postdoctoral Professional Development Program.

PI and affiliation: Christian Engelmann, ORNL


Team: R. A. Ashraf, R. Gioiosa (Pacific Northwest National Laboratory - PNNL), G. Kestor (PNNL), and R. F. DeMara (University of Central Florida).

Overview:

The strict power efficiency required to achieve exascale systems will result in an increase in the number of detected and undetected transient errors, which would directly affect the vulnerability of High-Performance Computing applications executed on these systems. Among the various hardware and software components that affect system resiliency, the impact of compiler optimizations on application vulnerability has not been widely investigated. This is significant since compiler optimizations are one of the most successful ways to improve performance with relatively low effort from the programmer. We analyze the tradeoffs between performance and vulnerability for multiple HPC applications compiled using standardized optimization levels. Results for these applications demonstrate that highly-optimized code is generally more vulnerable than un-optimized code. Increasing optimization levels can drastically improve performance as expected, however, it is observed that certain cases of optimization only provide marginal benefits in application performance yet considerably increase application vulnerability. In this work, we also define software attributes which are effected by changing compiler optimization levels, and contribute to increase in application vulnerability. For example, one of the attribute captures the probability of a faulty value to propagate to memory and corrupt the memory state of other processes in a parallel environment. Overall, our work shows that compiler cost functions should account for vulnerability in addition to standard metrics of code size and performance, and the importance of identifying important software attributes for this purpose.