Havens: Explicit Reliable Memory Regions for HPC Applications

Achievement: Developed a memory management method based on the notion of memory regions that prioritizes resilience and locality for high-performance computing (HPC) applications. Defined an abstract interface and developed a library-based proof-of-concept prototype implementation. Demonstrated its viability with a conjugate gradient solver and memory regions with different resilience against memory errors. Created programming language annotations to make a program's Havens more explicit and to easily incorporate resilience capabilities in HPC application codes.

Significance and Impact: Future HPC systems will contain more complex and denser memory hierarchies, and utilize more diverse memory technologies. Efficiently managing resilience against memory errors is a key challenge for the next generation of systems. Through language constructs for Haven-based memory management, HPC applications are able to explicitly manage the locality and resilience of program objects and computations.

Research Details:
• Developed a model for memory management for HPC applications that prioritizes resilience and locality of program objects.
• Havens extend the notion of region-based memory allocation method to permit HPC applications to explicitly manage the resilience and locality of reference of their application's data.
• Developed specification of Havens and designed an initial abstract interface (Figure 1).
• Prototyped a library implementation with a lightweight parity-based memory error detection and correction technique for each Haven.
• Developed a realistic proposal for adding language support for Havens to mainstream HPC languages.
• Defined type annotations, which enable static encoding of the program object's allocation and deallocation into the robust regions.
• Specified application-driven models of resilience that are made possible by the structured use of Havens (Figure 2).
• Demonstrated the model of selective reliability for program objects in a conjugate gradient solver.
• Created optimizations for parity-based protection scheme based on static type annotations.
• Investigated how the specification of the resilience of the individual program objects using these static annotations affects their fault coverage and performance during application execution.

Sponsor/Facility: This work was sponsored by the US Department of Energy, Office of Science, Early Career Research Program.

PI and affiliation: Christian Engelmann, CSMD – Oak Ridge National Laboratory

Team: S. Hukerikar (ORNL), and C. Engelmann (ORNL)

Publications:

Figure 1: Havens prototype architecture

Figure 2: Application reliability models using Havens
Overview:
Havens provide robust memory regions in which program objects may be allocated. A memory region is protected by a predefined error detection and/or correction scheme that is agnostic to the algorithm features or structure of the data. This approach to memory management enables separation between the memory allocation and the implementation of the robustness scheme. Havens enable applications to exert fine-grained control on the resilience properties of individual program objects. Since different Havens may have varying guarantees of reliability, based on the strength of the protection mechanism and its performance overhead, object placement in Havens may be driven by the trade-off between criticality of the object to program correctness and the associated overhead.

The language support for Havens offers explicit, convenient, sound and scalable control over Havens. The typing system enables application programmers to statically encode memory management decisions on the basis of their understanding of their application resilience requirements. This enables 3 different application-level reliability models: (1) selective reliability with Havens providing selective regions of program memory with comprehensive error protection, (2) specialized reliability with Havens providing different protection schemes based on application resilience and performance needs, and (3) dynamic reliability with Havens being enabled/disabled on demand during application execution.

Through accelerated fault injection experiments on a preconditioned conjugate gradient (PCG) solver (Figure 3(a)), we observed that the allocation of the static state into Havens yields the highest rate of successful completion. Also, providing the highest fault protection coverage does not yield a comparable increase in application resilience. Therefore, imposing the selective reliability model using Havens on the PCG solver provides the right balance between application resilience and performance overhead (Figure 3(b)).