ASTEROID - Analyzable, Resilient Real-Time Operating System Design

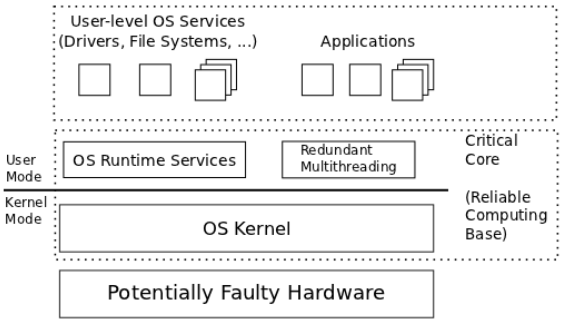
Björn Döbel, Hermann Härtig
TU Dresden
{doebel,haertig}@tudos.org

Philip Axer, Rolf Ernst
TU Braunschweig
{axer,ernst}@ida.ing.tu-bs.de

1 Introduction

The operating system (OS) plays a key role in any complex computing system. A current OS supporting software integration with memory management and virtualization contains several core functions that depend on error free hardware (HW). Errors in these functions quickly and irreversibly propagate through the system making it virtually impossible to recover from a function failure. Other OS functions can recover from failures with appropriate mechanisms. The ASTEROID project develops an OS and HW mechanisms that utilize the HW and communication resources of a many–core system to efficiently provide the required dependability. The goals are to (1) identify the critical core functionality, (2) to minimize the hardware and software resources needed for the core, (3) to establish interfaces and signalling between HW, OS, and applications so as to provide system integrity which shall be guaranteed by a corresponding formal safety analysis, and (4) to extend the underlying HW architecture to provide the necessary fault handling mechanisms.

2 System Overview

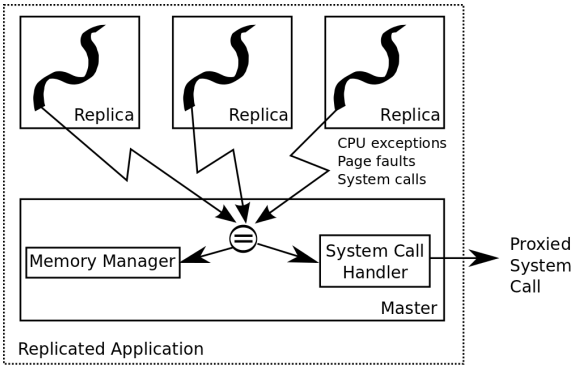


One source for transient errors are single-event upsets (SEUs), which are especially hard to detect and correct unless special techniques are applied. Without additional effort, at least some SEUs will not be masked and hence need to be handled by system software. For this purpose, ASTEROID uses the inherent redundancy of multi- and many–core hardware. Certain parts of the system, such as memory-based structures (e.g. MMU, cache, main memory) can easily be protected by error correcting codes and are therefore not in the scope of ASTEROID.

The goal of ASTEROID is to provide sufficient reliability for user-level application software. To this end, a critical core of operating system components is required to provide correct execution. Using a microkernel-based operating system, we can remove most of the non-critical OS functionality (device drivers, file systems, ...) from the core into user-space application where they can benefit from the OS's reliability mechanisms.

3 Operating System Support for Redundant Multithreading

The ASTEROID OS provides redundant multithreading as an operating system service. Unmodified applications are transparently replicated by the OS at the binary level, hence not requiring any source code access. Replicas execute on different hardware cores in a loosely coupled way. MMU-based address space isolation is used to protect fault propagation between replicas.



Replica threads are managed by a master process. This process compares replica states, proxies system calls, and makes sure that all replicas perceive the same inputs at exactly the same point in time. To reduce runtime overhead, these comparisons are only performend when the application externalizes state, e.g., by raising CPU faults or performing system calls.

The master supports efficient handling of memory. Replicas work on private copies of application-private memory in order to reduce runtime overhead and to avoid having to trap&emulate every memory access. Special handling for memory regions shared with external applications is provided. The complete implementation of the master process requires less than 3,000 lines of C/C++ code.

4 Execution Signatures for Efficient Integrity Checks

Checks on system calls and interrupts alone may lead to an increased error detection latency because the time between system calls may be arbitrarily long. Additionally, dormant errors such as errors in yet unused registers might not be detected on time. In order to decrease this latency, we propose to use fingerprints of the execution stream. The data and instruction streams are hashed and can be used as another source for voting. This is an especially cheap technique to reduce bandwith and comparision overhead.