An Operating System Framework for Transparent Service Resiliency

Mark Aiken
Microsoft Research
maiken@microsoft.com

Hiroo Ishikawa
Waseda University
ishikawa@cs.waseda.ac.jp

The Singularity project at Microsoft Research is a cross-discipline effort to construct reliable systems by applying recent innovations in systems design, safe programming languages, and tools. The project's main artifact, Singularity, is an operating system written almost entirely in safe languages, and meant to serve as a laboratory for developing new techniques and tools for ensuring system reliability and correctness.

Recent work within this project has focused on building facilities to support automatic and transparent resiliency of system services, and a framework to facilitate the development of such services. "Resiliency", here, means the ability to recover from failures by restarting the offending service. "Transparent" means that the recovery process is undetectable by client applications interacting with the service. We believe this approach can provide the means for recovering from bugs whose manifestation depends on events that are nondeterministic from the perspective of the service, including race conditions sensitive to specific interleavings of concurrent operations, and bugs on error paths when processing messages received from other components.

In our approach, a tool mechanically generates a companion component to each resilient service, based on a specification of each service's interaction with client applications, along with resiliency-specific annotations indicating the points at which the service's state should be captured. Each companion component intercepts and records its service's interactions with the system, coordinates the creation of periodic checkpoints that include a distillation of its service's state, and coordinates failure recovery by reconstituting its service from saved state and replaying recent messages. Authoring a resilient service requires conformance to various restrictions; the process is facilitated by a development framework.

Future generalizations include an expansion of scenarios in which recovery is possible by enabling "resiliency-aware" client applications to participate in the recovery process, enhancements to techniques for detecting service corruption or misbehavior, efficiency improvements, and extensions of the conceptual model to better support complete stacks of interoperating system services. We also intend to extend support to the application layer to achieve application resiliency.

Demonstration

A demonstration of the Singularity operating system running on commodity PC hardware, including an example resilient system service that can be crashed and transparently restarted without impairing multiple connected client sessions, will be provided.