Multiverse: Automatic Hybridization of Runtime Systems
Kyle C. Hale, Conor Hetland, and Peter Dinda | {kh, ch}@u.northwestern.edu

A Hybrid Runtime (HRT) is a transformation of a traditional parallel runtime into a specialized operating system kernel. HRTs enjoy unfettered access to the hardware and determine their own abstraction to that hardware.

The Hybrid Virtual Machine (HVM) makes it possible to create VMs that are internally partitioned between a "regular OS" (ROS) and an HRT. They allow the HRT to leverage legacy functionality inside the ROS, and they allow a user to easily create and launch HRTs from the ROS.

- We showed in previous work that by porting a legacy parallel runtime to an HRT environment, we can increase the performance of a real parallel runtime system by as much as 40% [2, 3] and the HRT is composed of the runtime and a thin kernel framework layer called an Aerokernel.
- Aerokernels are designed to be simple, lightweight, and very fast. We designed and implemented the Nautilus Aerokernel, which is used in conjunction with Multiverse.

**Why Automatic Hybridization?**

- HRTs can be very fast, but they require a manual port to kernel mode. This requires domain knowledge at the level of a runtime developer and at the level of a kernel developer.
- Even for an experienced kernel developer, porting a complex parallel runtime to kernel-mode is an error-prone process. Porting can be difficult and laborious!

**Building an Aerokernel to support a parallel runtime system (manual port to HRT)**

- Much of this functionality is not on critical path!

**References**


**Multiverse**

- In Multiverse, the runtime begins execution in the ROS. The runtime creates an HRT context through either explicit or implicit invocations.
- Once an HRT context is created, the system is in a state of split execution.
- During split execution, exceptional events on the HRT side (page faults, system calls, and some others) are forwarded to the ROS.
- With Multiverse, when a new HRT context is created, the Aerokernel is booted transparently on a remote set of cores.
- The Aerokernel binary is included in the runtime executable when compiled with our toolchain.
- The boot initialization is requested by the Multiverse runtime layer on the ROS side.

**Reducing Forwarded Events**

- These bars show the benchmarks that perform worse with Multiverse initially.
- We introduce a change to the runtime that eagerly faults in pages when mapping large chunks of memory.
- This reduces the occurrence of page faults, which turn reduces the number of events forwarded from HRT to ROS.
- Performance of the hybridized version of Racket is now better than virtual.
- The point of this exercise is to show that the overheads of Multiverse can be eliminated by reducing the number of forwarded events.

**Summary**

- We introduced Multiverse, a system that automatically hybridizes existing runtime systems.
- Runtime developers rebuild their system with our toolchain. It can then operate in a state of split execution, where most of the execution occurs in an accelerated, HRT environment.
- Multiverse adds little to no overhead, allowing the developer to start with a working system in kernel mode. The developer can then incrementally port legacy functionality to the HRT, reducing the number of events forwarded to the ROS.

**Aerokernel boot process**

- Each HRT execution context is paired with a partner thread on the ROS, which handles events forwarded over event channels. HRT contexts with their partner threads comprise execution groups.
- Nested threads share event channels with their parent.
- HRT contexts are (by default) created whenever a new pfthread is created in the runtime.