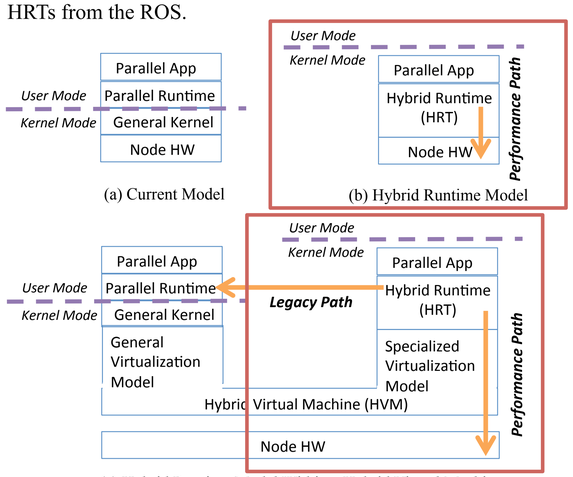
# Multiverse: Automatic Hybridization of Runtime Systems

Kyle C. Hale, Conor Hetland, and Peter Dinda | {kh, ch}@u.northwestern.edu, pdinda@northwestern.edu


## Hybrid Runtimes

*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 abstractions 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.



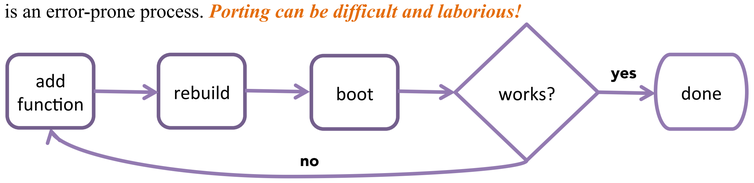

**fast path on HW**

**fast path under virtualization**

- 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]
- The HRT is composed of the runtime and a thin kernel framework layer called an ***Aerokernel***
- Aerokernels are designed to be simple, light-weight, 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!

***Need an easier way to go from legacy runtime system to HRT+HVM-capable runtime***

## References

[1] K. Hale, C. Hetland, and P. Dinda. *Automatic Hybridization of Runtime Systems*. In Proceedings of the 25th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC '16).

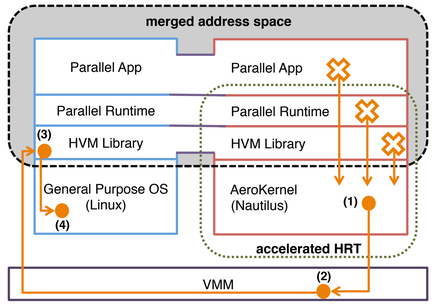
[2] K. Hale and P. Dinda. *Enabling Hybrid Parallel Runtimes Through Kernel and Virtualization Support*. In Proceedings of the 12th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments (VEE '16).

[3] K. Hale and P. Dinda. *A Case for Transforming Parallel Runtimes into Operating System Kernels*. In Proceedings of the 24th International ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC '15).

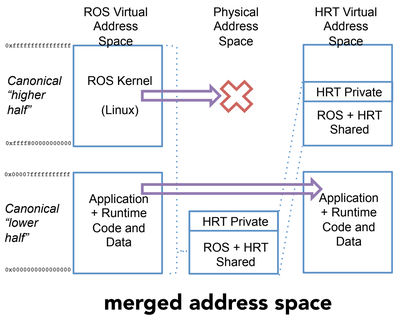
[4] J. Lange, P. Dinda, K. Hale, L. Xia. *An Introduction to the Palacios Virtual Machine Monitor Release 1.3*. Tech. Rep. NWU-EECS-11-10, Dept. of EECS, Northwestern Univ. (2011).

## 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



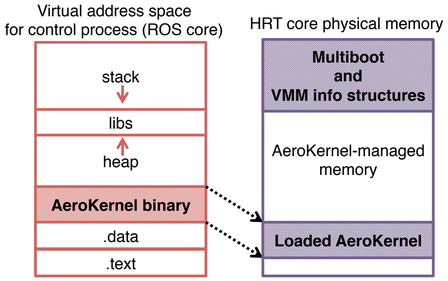

**split-execution in Multiverse**



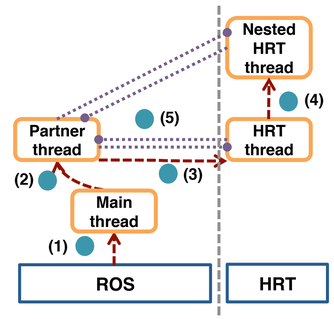

**merged address space**

- Merged address space allows HRT to ***leverage code/data mapped into the ROS virtual address space***
- We can, for example, use shared user-space libraries in the HRT that are mapped into the ROS process without implementing dynamic linking functionality in the Aerokernel
- The HRT can operate on data structures that have been constructed in the ROS
- Higher-half addresses (where the kernel code/data is mapped) are distinct for ROS and HRT

- With Multiverse, when a new HRT context is created, ***the Aerokernel is booted transparently on a remote set of cores***
- **Right**: The Aerokernel binary is included in the runtime's executable when compiled with our toolchain
- The boot initialization is requested by the Multiverse runtime layer on the ROS side




**Aerokernel boot process**

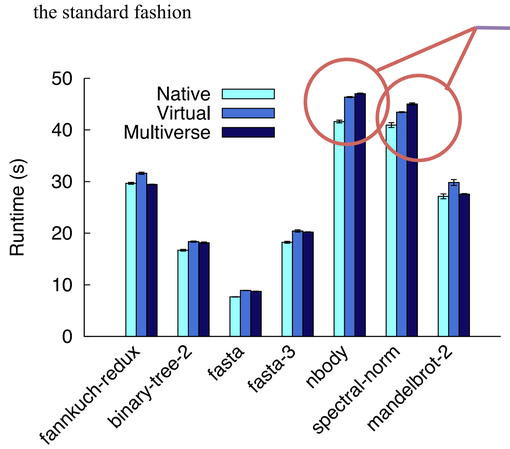



**interactions within an execution group**

- 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 pthread is created in the runtime

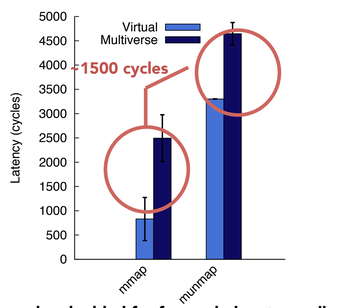
## Small Performance Overhead

- **Racket** is the most widely used dialect of Scheme
- Includes challenging features typical of a dynamic, high-level language. ***Many make heavy use of Linux ABI***: system calls, memory mapping, processes, threads, signals, etc.
- We automatically hybridize Racket with Multiverse. The user can interact with the Racket REPL in the standard fashion



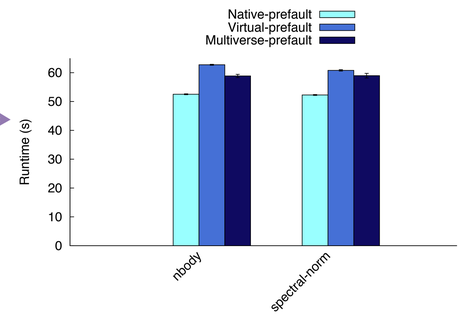

**Left:** Performance of hybridized Racket (with Multiverse) for a set of benchmarks from the Language Benchmark Game compared to Racket in a VM and Racket running on native Linux. ***Overheads are very small***. In all but two cases, the light-weight environment provided by the HRT actually increases performance over Virtual.

**Right:** The primary source of overhead in Multiverse comes from forwarded events. The two benchmarks above that are made slower from overhead incur most of it from page faults. The figure on the right shows that the ***overheads for typical forwarded events are roughly 1500 cycles*** for each event.




**overhead added for forwarded system calls**

## Reducing Forwarded Events




- These bars show the benchmarks that perform *worse* with Multiverse initially
- We introduce a change to the Racket runtime that eagerly faults in pages when mapping large chunks of memory
- This reduces the occurrence of page faults, which in 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

## Acknowledgements

*This project is made possible by support from the United States National Science Foundation through grant CCF-1533560 and from Sandia National Laboratories through the Hobbes Project, which is funded by the 2013 Exascale Operating and Runtime Systems Program under the Office of Advanced Scientific Computing Research in the United States Department Of Energy's Office of Science.*