MULTIVERSE
Automatic Hybridization of Runtime Systems

Kyle C. Hale, Conor Hetland, Peter Dinda
HYBRID PARALLEL RUNTIMES

LONG-TERM GOAL:
let’s reimagine systems S/W stack for parallel runtimes (esp. for high-level parallel languages)

why in the world?

- imposed abstractions, hides HW capabilities
- avoid duplicated functionality in runtime system and OS
- explore new ways of doing things!
A THOUGHT EXPERIMENT
2 ASSUMPTIONS

1. YOU ARE A **HARDCORE OS HACKER**
   - expert **parallel programmer**
   - elite **performance engineer**
   - you can squash **bugs** like no one’s business
   - you have **masochistic tendencies**
2 ASSUMPTIONS

2. YOU HAVE **INFINITE TIME** ON YOUR HANDS
NOW, GO BUILD A PARALLEL RUNTIME SYSTEM

*on raw hardware
HOW **DIFFERENT** IS YOUR RUNTIME FROM A TYPICAL OS KERNEL?

[Hale et al., HPDC ‘15]

[Hale et al., VEE ‘16]
PROBLEM:

you’ve already built a runtime for Linux
building it from scratch is hard
Linux user-space code specialized OS kernel
OKAY, LESS AMBITIOUS

I’ll give you a kernel framework

Now you just have to port, probably add some functionality
instead of starting from scratch, we can port to kernel mode:

DIFFICULT

TIME-CONSUMING

ERROR-PRONE
development cycle:

do {
    ADD FUNCTION
    REBUILD
    BOOT specialized kernel
}
while (kernel falls over)
FURTHERMORE

much of the functionality is

NOT ON THE CRITICAL PATH
MULTIVERSE
C/C++ source tree for runtime system

point Multiverse to runtime’s source tree

rebuild

Automatic Hybridization
run it (boots as kernel)

identify hotspots

bring functionality into your kernel
leverage general-purpose OS for legacy functionality
MULTIVERSE IMPLEMENTS AUTOMATIC HYBRIDIZATION

Universe 1
- your runtime system
- Linux

Universe 2
- your runtime system
- Specialized OS environment

Hardware (Virtual or Physical)
ON TOP OF VIRTUALIZATION

- your runtime system
- Multiverse runtime layer
- Linux

events: page faults, system calls, exceptions

- thread_create()
- your runtime system
- specialized OS

Hybrid Virtual Machine
FREEBIE: WORKS WITH OUT OF THE BOX PTHREADS APPS
FREEBIE: WORKS WITH OUT OF THE BOX PTHREADS APPS

- pthread app
- Multiverse runtime layer
- Linux

Hybrid Virtual Machine

- low noise environment
- specialized OS
RACKET

most widely used Scheme implementation runtime system: 800K lines of code complex, JIT-based runtime, managed memory, garbage collection, etc. complexity makes it good option for testing Multiverse
bench-write.out    go         mracket-GOLD
binary-tree-2.rkt   intsum-native multiverse.log
bytes               isn         nbody.rkt
collects            lgn-hpcg    racket
doall.sh            lgo         results
doruns.sh           lost+found spectral-norm.rktfannkuch-redux.rkt  lpm         test.out
fasta-3.rkt         lpn         test.t
fasta.rkt           mandelbrot-2.rkt

# 1s
# ls
bench-write.out  go  mracket-GOLD
binary-tree-2.rkt  intsum-native  multiverse-racket
bytes  isn  multiverse.log
collects  ism  nbody.rkt
doall.sh  lgn-hpcg  racket
doruns.sh  lgo  results
fannkuch-redux.rkt  lost+found  spectral-norm.rkt
fasta-3.rkt  lpm  test.out
fasta.rkt  lpn  test.t

#
LOW OVERHEADS

![Runtime Graph]

- fannkuch-redux
- binary-tree-2
- fasta
- fasta-3
- nbody
- spectral-norm
- mandelbrot-2
MULTIVERSE

generate a specialized OS kernel (HRT) from a Linux user-space app/runtime system by rebuilding

hybridized runtime is \textit{bridged with Linux} to provide legacy functionality (e.g. system calls)

very slight overheads (100s of ns) for forwarded events, little effect on performance overall
THANKS

my website: http://halek.co

Nautilus Aerokernel Framework: http://nautilus.halek.co

Prescience Lab: presciencelab.org

BACKUPS