Unikernel Linux

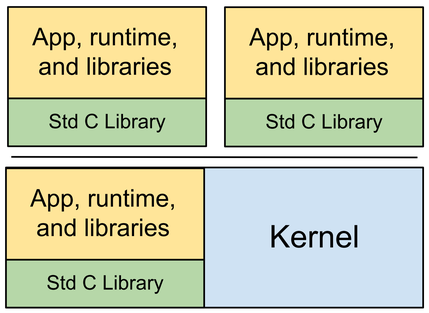
Ali Raza *, Thomas Unger *, Matthew Boyd *, Eric Munson *, Parul Sohal *, Ulrich Drepper ◇, Richard Jones ◇, Daniel Bristot de Olivera ◇, Larry Woodman ◇, Renato Mancuso *, Jonathan Appavoo *, Orran Krieger *

*Boston University, ◇Red Hat

Problem:
General Purpose Operating Systems (GPOS) have high overhead for modern application deployments, but unikernels require significant developer effort and sacrifice common admin tools.

Solution:
Unikernel Linux (UKL) offers a configuration spectrum between GPOSs and traditional Unikernels. Starting by linking a privileged application with the kernel, then exploring optimizations enabled by this linkage.

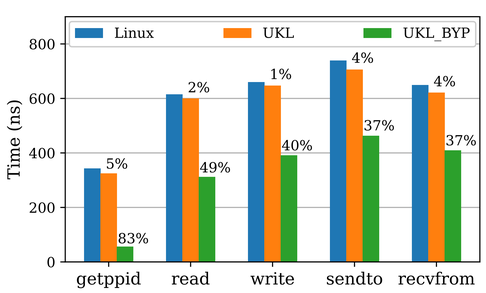
Architecture:
UKL differs from a unikernel by maintaining the process abstraction and allowing a standard user space to coexist with the linked application.



Implementation:
- Base UKL is implemented in ~500 lines of code and being prepared for submission to the upstream Linux kernel.
- Full list of currently supported optimizations is ~1k lines of code.
- Each optimization is independently configurable, allowing developers to trade between GPOS compatibility and unikernel performance in discreet steps.

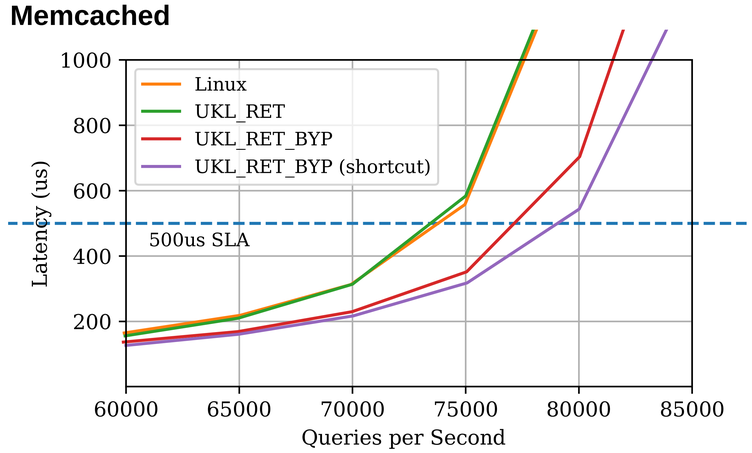
Performance:
Micro-benchmarks demonstrate savings on system calls.

UKL configurations are base (UKL) and skip syscall entry and exit code for 9 of 10 calls (UKL_BYP).



Application Performance:

Memcached



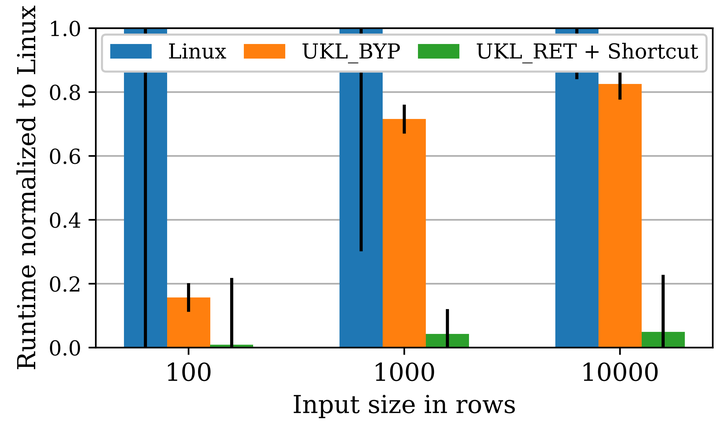
Using the ability to call directly into the kernel internals, gives an 8% improvement in queries per second under the 500us SLA.

Redis

System	99% tail (ms)	Tail improv. (%)	Throughput (Kb/s)	T-put improv. (%)
Linux	3.26	--	6375.20	--
UKL_BYP	2.91	11%	7154.68	12%
UKL_BYP w/Shortcut	2.54	22%	8022.54	26%

Redis saw good inprovements with no code modifications (11% in tail latency, 12% in throughput). However, making redis interact with the TCP stack directly, tail latency improved by 22% and throughput by 26%.

Secrecy



Secrecy is a latency sensitive, multi-party computation application. By accelerating both send and receive paths for all participants we see a 100x improvement in run times for the GROUP-BY operator. Error bars are the coefficients of variation.

Results:
- Performance improves with base UKL for most applications
- Significant improvements come with using optimization techniques that require more developer effort.
- It's "just Linux" so tools like ftrace, top, and perf still work!

For more information contact:
Ali Raza <aliraza@bu.edu>
Eric Munson <munsoner@bu.edu>
Richard Jones <rjones@redhat.com>

Visit our project site at https://github.com/unikernelLinux/ukl

Or with the QR code

Red Hat Research

BOSTON UNIVERSITY