Unikraft – Crafting Unikernels

Simon Kuenzer, Felipe Huici, Florian Schmidt
<firstname.lastname>@neclab.eu
SYSML Group
NEC Laboratories Europe
http://sysml.neclab.eu
Advantages and the Problem with Specialization

Standard OS/VM/container image:
- **lots** of unnecessary code
- **lots** of overhead!

Specialized image: only what’s needed is there **but lots** of development time! (have to change code **by hand**)

![Diagram showing differences between standard and specialized images](image)
In a perfect world...

- We would have a menu of libraries for all possible components applications might need.
- We would be able to use that menu to select *only* the functionality needed (and possibly automate the selection process).
- A system would *automatically* build a lean, high performance image for the application we’re interested in and the platforms we care for.

Unikraft is precisely this system!
Standard operating systems are **monolithic**: they are not modular so it’s *not* possible to separate their parts.

<table>
<thead>
<tr>
<th>Application(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>network stack</td>
</tr>
<tr>
<td>profiling</td>
</tr>
<tr>
<td>memory allocator</td>
</tr>
<tr>
<td>filesystem</td>
</tr>
<tr>
<td>timers</td>
</tr>
<tr>
<td>scheduler</td>
</tr>
<tr>
<td>drivers</td>
</tr>
</tbody>
</table>
Could we decompose, i.e., break apart an operating system?
Could we decompose, i.e., break apart an operating system?

Once decomposed, we can pick and choose which parts/libraries we actually need for our application!
The Unikraft Library and Build System

1. SELECT APP
   - mysql
   - QEMU
   - mailman
   - nginx
   - snort
   - memcached
   - myapp

2. CUSTOM ABI/API
   - archlibpool
     - platformlibpool
       - libbareplat.o
       - liblinuxuplat.o
       - libxenplat.o
       - libkvmplat.o
     - archlibpool
       - libx86_64arch.o
       - libarm32arch.o
       - libmipsarch.o

3. SELECT&CONFIG LIBS
   - network stack
     - liblwip.o
     - libtcpip.o
     - libhttp.o
   - drivers
     - libconsole.o
     - libixgbe.o
     - libnetfront.o
   - filesystems
     - libvfs.o
     - libfat.o
     - libext3.o
   - memory allocators
     - libbuddy.o
     - libheap.o
     - libmempool.o
   - schedulers
     - libcoop.o
     - libpreempt.o
     - librt.o
   - runtimes
     - libocaml.o
     - libpython.o
     - liberlang.o
   - standard libs
     - libc.o
     - libnewlibc.o
     - libopenssl.o
   - debug&profiling
     - libgdb.o
     - libucdebug.o
     - libperf.o

4. BUILD
   - unikernel binaries
     -unikraft_bare_x86_64
     -unikraft_bare_ARM32
     -unikraft_bare_MIPS
     -unikraft_xen_x86_64
     -unikraft_xen_ARM32
     -unikraft_xen_MIPS
     -unikraft_kvm_x86_64
     -unikraft_kvm_ARM32
     -unikraft_kvm_MIPS
     -unikraft_linuxu
Building a Specialized Image in One Minute

- Type “make menuconfig”
- Choose options in the menu that you want for your application
- Choose your target platform(s) (e.g., Xen, KVM, Linux, baremetal)
- Save config and type “make”
An Unikraft Image Example

Xen PV x86_64 binary
Compiles to a 32.7kB image

unikraft_xen-x86_64.o (50,2kB)

Boots and prints message to debug console (with min. 208kB RAM)
Potential Unikraft-built Systems

Specialized Python images for Xen, KVM and ARM, x86_64

Verticals:
- IoT gateways
- Smart city gateways
- Cloud computing platforms (e.g., AWS Lambda)
# Potential Unikraft-built Systems

## Specialized NFV image for KVM on x86_64

<table>
<thead>
<tr>
<th>libdpdk.o</th>
<th>liballocbuddy.o</th>
</tr>
</thead>
<tbody>
<tr>
<td>libschedcoop.o</td>
<td>libconsole.o</td>
</tr>
<tr>
<td>libkvmlat.o</td>
<td>libx86_64arch.o</td>
</tr>
</tbody>
</table>

### Verticals:
- vCPE
- vRouters
- vBRAS
- Your network application here!

---

© NEC Corporation 2017