**What is JOVE?**

- JIT Compilation of a Commodity OS in a Virtualized Environment
- Extension of Secure Virtual Architecture (SVA) work from LLVM Researchers
- Foundation for new work

<table>
<thead>
<tr>
<th>Commodity OS</th>
<th>Virtual ISA</th>
</tr>
</thead>
<tbody>
<tr>
<td>JIT Compiler + VMM</td>
<td>Physical ISA</td>
</tr>
</tbody>
</table>

**Why is this important?**

- Improvements to SVA
  - Provides Trusted Boot mechanism to SVA
  - Can boot arbitrary bytecode kernels and apply safety to them
  - Kernel module loading safety
- Bring JIT benefits that userspace enjoys to the kernel
  - Foundation for new applications with exciting potential
  - Performance
  - Security
  - Portability
- Provides a single general mechanism, namely a JIT for
  - Instrumentation
  - Profiling
  - Patching

**How is this done today?**

**It’s not**

**System Description**

**JOVE Boot Process**

- Guest
  - Bootstrap Code
- JIT Compiler + VMM
- Hardware

**Contents of real kernel here as bytecode**

**Lazy Compilation Callback Hypercall**

- Builds functions as the Guest encounters and requests them. Rewrites guest code so doesn’t request function again (faster).

**Function stubs and the lazyCC call**

**Performance**

94% Bandwidth

17.5% Latency Overhead

**Future Work**

- Dynamic Optimizations
  - Online profiling
  - Debugging/tracing applications
  - Dynamic optimizations based on profile feedback
- Code specialization
  - Take advantage of run-time information
  - Especially useful in e.g. drivers
- Architecture-specific optimizations
  - Take advantage of hardware features that weren’t known at build time
- Replace Interpreters within the kernel
  - e.g. Packet filter interpreter could be replaced with JIT calls to emit native code
- Security Applications
  - Online patching security vulnerabilities
  - No need to reboot
  - Full source not needed, can replace an entire function using only compiled replacement
- Portability
  - Bytecode portability
  - Because code is generated at run-time, there is the potential to have a cross-platform kernel
  - Cross-platform drivers
  - Given stable ABI a single driver binary could work on many platforms