Exokernel, IX, Arrakis

Antoine and Eric
Overview

1. Exokernel
2. IX
3. Arrakis
4. Discussion
OS: All about abstractions

● Traditional OS provides:
  ○ Protection + resource management + HW abstractions

● Generally there is more than one way to abstract
  ○ OS needs to pick one
  ○ Different abstractions have different performance
Abstractions considered harmful

Not all applications want the same abstractions

- Performance differences
- Hiding too much information
  - can make it difficult for the application to implement functionality (e.g. databases, user space threads)
- Changing “the one” abstraction can be difficult to impossible
Exokernel

- Kernel only implements protection
- Abstraction is implemented in user space
- Expose:
  - Hardware (securely)
  - Physical Names
  - Allocation
  - Revocation
Library Operating Systems

- Implement abstractions in user space
- Application can choose which library to use
  - Abstractions can be tailored to application
  - No need for general purpose implementations
- Can avoid many kernel crossings: function calls instead of system calls
Secure Bindings

- Separate authorization from use
- Only perform authorization check at bind time
- At access time only simple access check required
- Implement using: hardware, caching, and downloading code
“Downloading” code into kernel

- Allows pieces of code to be pushed into kernel
- Not specific to an application
- Helpful for things that need to go fast like packet filters

c.f. Berkeley Packet Filter in Linux
Linux I/O Performance

<table>
<thead>
<tr>
<th>Operation</th>
<th>HW %</th>
<th>Kernel %</th>
<th>App %</th>
<th>Time (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>18%</td>
<td>62%</td>
<td>20%</td>
<td>9</td>
</tr>
<tr>
<td>SET</td>
<td>13%</td>
<td>84%</td>
<td>3%</td>
<td>163</td>
</tr>
</tbody>
</table>

Kernel Features:
- API
- Naming
- Access control
- I/O Processing
- Protection
- Multiplexing
- Resource limits
- I/O Scheduling
- Copying

Hardware:
- 10G NIC: 2 us / 1KB packet
- RAID Storage: 25 us / 1KB write
Stolen IX slides
Arrakis

- Directly map vNIC into applications
- Implement network stack as a library
- Use hardware features for protection and demultiplexing to multiple applications:
  - Hardware I/O virtualization: SR-IOV, IOMMU
  - NIC: packet filters, rate-limiting
Arrakis: Control/Data plane split

• Kernel is control plane, data plane is fully in user space
• Control plane not on the critical path
  ○ Packets sent/received directly from user space
• Invoke control plane only infrequently
  ○ Changing packet filters/rate limits
  ○ Analogous to Exokernel’s secure bindings
Arrakis I/O Architecture

Control Plane

Kernel
- Naming
- Access control
- Resource limits

Data Plane

Redis
- API
- I/O Processing

I/O Devices
- Protection
- Multiplexing
- I/O Scheduling

Data Path
Arrakis: Storage stack

- Same idea can be applied to storage
- Direct access to storage controller for applications
- Application-specific persistent data structures instead of general purpose FS
- IPC interface to allow indirect access through VFS.
Arrakis Performance

- Reduced (in-memory) GET latency by 65%

<table>
<thead>
<tr>
<th></th>
<th>HW 18%</th>
<th>Kernel 62%</th>
<th>App 20%</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td></td>
<td></td>
<td></td>
<td>9 us</td>
</tr>
<tr>
<td>Arrakis</td>
<td>HW 33%</td>
<td>libIO 35%</td>
<td>App 32%</td>
<td>4 us</td>
</tr>
</tbody>
</table>

- Reduced (persistent) SET latency by 81%

<table>
<thead>
<tr>
<th></th>
<th>HW 13%</th>
<th>Kernel 84%</th>
<th>App 3%</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux (ext4)</td>
<td></td>
<td></td>
<td></td>
<td>163 us</td>
</tr>
<tr>
<td>Arrakis</td>
<td>HW 77%</td>
<td>libIO 7%</td>
<td>App 15%</td>
<td>31 us</td>
</tr>
</tbody>
</table>
IX vs Arrakis

- Do we need to protect the network stack from the application?
  - To what degree is the OS responsible for correct protocol implementation?
  - Possible problems: congestion control, others?
  - How does this differ from the cloud setting?
  - Would more hardware support help?
IX, Arrakis applicability

- Both are primarily targeted at data centers
- Are there other settings where they could be useful?
  - Mobile? Desktops?
Exokernel

- Interesting ideas for applications?
- VMMs vs Exokernel?
- What challenges prevent this from being the standard kernel structure today?
  - What are possible solutions to those challenges?
Specialization

- Both IX and Arrakis provide POSIX-compatible interfaces that come with some performance cost
  - -> Inherent complexity vs performance trade-off