A Way Forward: Enabling Operating System Innovation in the Cloud

Dan Schatzberg, James Cadden, Orran Krieger, Jonathan Appavoo
Boston University
There has been a lot of innovation in cloud computing
In contrast, there has been relatively little innovation in the operating system
Do operating systems matter?
Do operating systems matter? Yes!

Memcached Latency

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
<th>Mean latency (μs)</th>
<th>99 %ile latency (μs)</th>
<th>Overall share</th>
</tr>
</thead>
<tbody>
<tr>
<td>DC Fabric</td>
<td>Propagation delay</td>
<td>&lt; 1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Single Switch</td>
<td>1-4</td>
<td>40-60</td>
<td>1%</td>
</tr>
<tr>
<td></td>
<td>Network Path†</td>
<td>6</td>
<td>150</td>
<td>7%</td>
</tr>
<tr>
<td>Endhost</td>
<td>Net. serialization</td>
<td>1.3</td>
<td>1.3</td>
<td>1.4%</td>
</tr>
<tr>
<td></td>
<td>DMA</td>
<td>2.6</td>
<td>2.6</td>
<td>3%</td>
</tr>
<tr>
<td></td>
<td><strong>Kernel</strong> (incl. lock contention)</td>
<td><strong>76</strong></td>
<td><strong>1200-2500</strong></td>
<td><strong>86-95 %</strong></td>
</tr>
<tr>
<td>Application</td>
<td>Application*</td>
<td>2</td>
<td>3</td>
<td>2%</td>
</tr>
<tr>
<td></td>
<td><strong>Total latency</strong></td>
<td>88</td>
<td>1356-2656</td>
<td>100%</td>
</tr>
</tbody>
</table>

Reproduced from Kapoor et al. Chronos predictable low latency for data center applications. SOCC 2012
How can we enable OS innovation in the cloud?
Modify Existing OS

Clean Slate
<table>
<thead>
<tr>
<th>Pros</th>
<th>Clean Slate</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Large software ecosystem</td>
<td></td>
</tr>
<tr>
<td>Modify Existing OS</td>
<td>Clean Slate</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td></td>
</tr>
<tr>
<td>✓ Large software ecosystem</td>
<td></td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td></td>
</tr>
<tr>
<td>- Large and complex</td>
<td></td>
</tr>
<tr>
<td>Modify Existing OS</td>
<td>Clean Slate</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------</td>
</tr>
<tr>
<td><strong>Pros</strong></td>
<td></td>
</tr>
<tr>
<td>✓ Large software ecosystem</td>
<td>✓ Complete control</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td></td>
</tr>
<tr>
<td>- Large and complex</td>
<td></td>
</tr>
</tbody>
</table>
Modify Existing OS versus Clean Slate

**Pros**
- ✓ Large software ecosystem
- ✓ Complete control

**Cons**
- - Large and complex
- - Need to reproduce legacy functionality
A Way Forward
Typical Cloud Web Application

MySQL
Java
Memcached

Apache

Client
Memcached

Typical Cloud Web Application

- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application
What is the role of the operating system?
Memcached

• Distributed application

• On demand provisioning

• Isolated (virtual) hardware for each application

Typical Cloud Web Application

The OS doesn’t need to:

1. Support multiple users
Memcached

• Distributed application
• On demand provisioning
• Isolated (virtual) hardware for each application

Typical Cloud Web Application

The OS doesn’t need to:
1. Support multiple users
2. Arbitrate and balance competitive resource usage
Memcached

- Distributed application
- On demand provisioning
- Isolated (virtual) hardware for each application

Typical Cloud Web Application

The OS doesn’t need to:
1. Support multiple users
2. Arbitrate and balance competitive resource usage
3. Use a symmetric structure to provide OS functionality
Typical Cloud Web Application

- MySQL
- Java
- Memcached
- Apache

Diagram showing the components of a typical cloud web application.
Typical Cloud Web Application

MySQL

Java

Memcached

Apache
**Typical Cloud Web Application**

- **MySQL**
- **Java**
- **Memcached**

**Library OS:** functionality linked into application address space
Typical Cloud Web Application

MySQL

Java

Memcached

Apache

World Wide Web
Typical Cloud Web Application

MySQL

Java

Memcached

Apache

World Wide Web
Typical Cloud Web Application

MySQL

Java

Memcached

Apache

Web Server
Typical Cloud Web Application

But what about

- logging
- management (ps, kill)
- configuration
- tooling
Typical Cloud Web Application

But what about

- logging
- management (ps, kill)
- configuration
- tooling

Offload non-performance critical functionality to general purpose OS
Typical Cloud Web Application

- MySQL
- Java
- Memcached

Apache

World Wide Web
Typical Cloud Web Application

MySQL
Java
Memcached

MultiLibOS

Apache
A MultiLibOS is a tightly integrated composition of general purpose operating systems and specialized library operating systems.
There are many different ways a MultiLibOS might be integrated
There are many different ways a MultiLibOS might be integrated

- Distributed Shared Memory
- Message Passing
- Distributed Namespace (9p)
The MultiLibOS allows a tradeoff between effort and utility
Lower Effort

Lower Utility

Higher Effort

Higher Utility

Existing General Purpose Systems
Typical Cloud Web Application

- MySQL
- Java
- Memcached
- Apache
Linux Memcached

- NIC Driver
- Network Processing
- Memcached

Interrupt
Linux Memcached

- NIC Driver
- Network Processing
- Memcached

- Interrupts Enabled
- Context Switch
- Schedule thread
- Interrupt
Linux Memcached

- Memcached
- Network Processing
- NIC Driver

- Interrupts Enabled
- Copy packet to userspace
- Context Switch
- Schedule thread

Interrupt
Linux Memcached

- Memcached
- Network Processing
- NIC Driver

- Copy packet to userspace
- Context Switch
- Validation, protection from DoS
- Schedule thread

Interrupts Enabled
Example Library OS Memcached

Memcached

Network Processing

NIC Driver

Interrupt
Example LibraryOS Memcached

Memcached

Network Processing

NIC Driver

Interrupt

Function Call
Example LibraryOS Memcached

Diagram showing the relationship between Memcached, Network Processing, NIC Driver, and interrupts. Interrupts are disabled below Network Processing, and function calls are indicated between the layers.
Example LibraryOS Memcached

- Memcached
- Network Processing
- NIC Driver

- Interrupts Disabled
- Zero Copy
- Function Call
- Function Call
- Interrupt
Example LibraryOS Memcached

- No virtual memory
- No complex scheduling
- Small system image
SageMath

- Open source mathematics environment (like Matlab, Mathematica, etc.)
- Python
- Incorporates many libraries
SageMath

- Open source mathematics environment (like Matlab, Mathematica, etc.)
- Python
- Incorporates many libraries
new Matrix(1000000, 1000000)
SageMath

\[
\begin{align*}
  f(x) & \quad f(x) & \quad f(x) & \quad f(x) & \quad f(x) & \quad f(x) & \quad f(x) & \quad f(x) \\
\end{align*}
\]

new Matrix(1000000, 1000000)
Focus on:
- I/O
- Elasticity
- Scalability
MultiLibOS Challenges

- Reuse vs. Specialization
- “Versionitis”
- Avoid building a new OS
Elastic Building Block Runtime (EbbRT)

- A MultiLibOS “toolkit” for elastic applications
- Components are
  - Efficient
  - Reusable
  - Customizable
Questions?

https://github.com/sesa/ebb3t

Dan Schatzberg, James Cadden, Orran Krieger, Jonathan Appavoo

dschatz@bu.edu   jmcadden@bu.edu   okrieg@bu.edu   jappavoo@bu.edu