Unikernel Linux (UKL)

Isaiah Stapleton
Software Engineer Intern
Research Team
What we’ll discuss today

- Brief Introduction
- Background
- Project Description
- Main Task
- Why this Work Matters
- Current Progress
- Side Task
- Problems & Solutions
- Next Steps
- Related Work
- Things I Learned
- Conclusion
- Q & A
Brief Introduction

- Born in Dover, NH
- Moved to Charleston, SC ~12 years ago
- Going into senior year at the College of Charleston
  - Computer Science Major & Data Science Minor
- Passionate about Linguistics (Language & Language Acquisition)
  - Currently working on one year research project designing a mobile language learning app utilizing comprehensible input
Background
What is a Unikernel?

A highly specialized OS that is designed to run a single privileged application that is linked with a kernel and executes in a single address space

- Only contains application code as well as all necessary operating system functions required by that application
Unikernel Architecture
What is the point of a Unikernel? What problem do they aim to solve?

- Virtual machines are often used to run only a single application belonging to a single user
- No need to support:
  - Isolation between different users
  - Thousands of drivers
  - Isolation between different processes
- Benefits:
  - Fast boot times
  - Increased performance
  - Increased security
Project Description
Unikernel Linux (UKL)

▸ Small patch to Linux and glibc which allows one to build many programs as Unikernels

▸ A system for applying Unikernel optimization techniques to the Linux Kernel

▸ Current Goals: Offer a selection of applications optimized using Unikernel techniques and working towards upstream submission of the UKL patch set
Why did I choose this project?
Main Task
Main Task

Port Nginx over to UKL and perform benchmark testing in order to compare results to the Lupine benchmark tests.
Why this Work Matters
Why?

In order to get our changes to the Linux kernel pushed upstream we have to show that our changes to these applications can make a significant difference in their performance.
Current Progress
Build Nginx from source

Figure out what dependencies it has (and their dependencies, all the way back to glibc)

Make sure we can build entire dependency tree tree and Nginx in a UKL compatible way
Current Progress

- Build Nginx in a UKL compatible way
- Boot Nginx as a UKL kernel
- Perform benchmark tests on UKL-Nginx
## Benchmark

### UKL-Nginx

<table>
<thead>
<tr>
<th>Document Path:</th>
<th>/index.html</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Length:</td>
<td>612 bytes</td>
</tr>
<tr>
<td>Concurrency Level:</td>
<td>100</td>
</tr>
<tr>
<td>Time taken for tests:</td>
<td>6.568 seconds</td>
</tr>
<tr>
<td>Complete requests:</td>
<td>100000</td>
</tr>
<tr>
<td>Failed requests:</td>
<td>0</td>
</tr>
<tr>
<td>Total transferred:</td>
<td>8450000 bytes</td>
</tr>
<tr>
<td>HTML transferred:</td>
<td>6120000 bytes</td>
</tr>
<tr>
<td>Requests per second:</td>
<td>15225.91 [#/sec] (mean)</td>
</tr>
<tr>
<td>Time per request:</td>
<td>6.568 [ms] (mean)</td>
</tr>
<tr>
<td>Time per request:</td>
<td>8.066 [ms] (mean, across all concurrent requests)</td>
</tr>
<tr>
<td>Transfer rate:</td>
<td>12564.35 [Kbytes/sec] received</td>
</tr>
</tbody>
</table>

### Lupine-Nginx

<table>
<thead>
<tr>
<th>Document Path:</th>
<th>/index.html</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Length:</td>
<td>612 bytes</td>
</tr>
<tr>
<td>Concurrency Level:</td>
<td>100</td>
</tr>
<tr>
<td>Time taken for tests:</td>
<td>9.119 seconds</td>
</tr>
<tr>
<td>Complete requests:</td>
<td>100000</td>
</tr>
<tr>
<td>Failed requests:</td>
<td>0</td>
</tr>
<tr>
<td>Total transferred:</td>
<td>9450000 bytes</td>
</tr>
<tr>
<td>HTML transferred:</td>
<td>6120000 bytes</td>
</tr>
<tr>
<td>Requests per second:</td>
<td>10965.75 [#/sec] (mean)</td>
</tr>
<tr>
<td>Time per request:</td>
<td>9.119 [ms] (mean)</td>
</tr>
<tr>
<td>Time per request:</td>
<td>9.001 [ms] (mean, across all concurrent requests)</td>
</tr>
<tr>
<td>Transfer rate:</td>
<td>9648.89 [Kbytes/sec] received</td>
</tr>
</tbody>
</table>
Side Task
Side Task

Implement TCP shortcuts in order to improve the performance of Nginx
Current Progress

Implement send shortcut

Implement receive shortcut
Problems & Solutions
## Task: Main Task

<table>
<thead>
<tr>
<th>Problem</th>
<th>Root Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open <code>/dev/null/ failed, no such file or directory</code></td>
<td><code>/dev</code> not populating when building UKL-Nginx</td>
<td>Richard Jones added changes to UKL to change the way applications are built</td>
</tr>
</tbody>
</table>
### Task: Main Task

<table>
<thead>
<tr>
<th>Problem</th>
<th>Root Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socketpair() failed while spawning worker process</td>
<td>A function in the Linux kernel (putuser) that writes values from kernel space into user space. If destination was found to be outside of user space the process fails / terminates</td>
<td>Modified Linux kernel to bypass this check</td>
</tr>
</tbody>
</table>
## Task: Side Task

<table>
<thead>
<tr>
<th>Problem</th>
<th>Root Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undefined reference to shortcut_tcp_send msg</td>
<td>Nginx failing to build, causing the entire build process to halt</td>
<td>Editing the Nginx Makefile to ignore failure to build Nginx in order to continue the build process</td>
</tr>
</tbody>
</table>
Task: Side Task

<table>
<thead>
<tr>
<th>Problem</th>
<th>Root Cause</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel panic during benchmark testing</td>
<td>???</td>
<td>???</td>
</tr>
</tbody>
</table>
Next Steps
Next Steps

- Get Lupine benchmark tests running on same hardware ✓
- Compare UKL-Nginx benchmark tests to the Lupine-Nginx benchmark tests ✓
- Further optimize UKL-Nginx ✓
Tools I Need to Learn

- Cscope
- GDB
- Vim
- C
Related Work
Related Work

- Linux Kernel Development Workshop
  - Jonathan Cameron

- Linux Kernel Development 3rd Edition by Robert Love
Things I learned
Things I Learned

- Kernel Development
- Open Source Development
- Git
- Presentation skills
- How and when to ask for help
- How to fight imposter syndrome
Conclusion
Thank you!

Technical Mentors: Eric Munson & Larry Woodman
Early Talent Mentor: Shamair Faison
Manager: Heidi Dempsey

Ali Raza, Richard Jones, and the rest of the UKL & research team
Questions?

Connect with me on LinkedIn