Ryan Sullivan U73687595
EC601
14 September 2022

Elevated memcached

Problem Statement:

A chronokernel is a tool developed by PhD. candidate Tommy Unger, and I have had the pleasure of working with him this past summer. This tool, simply put, allows the programmer to dynamically "elevate" and "lower" from user to supervisor mode execution and back, basically giving the programmer the full power of the kernel at any point in their program, on top of this power the programmer is still able to wield the full capabilities of their shared libraries. This allows for extremely interesting application optimizations: the one I am proposing to investigate is kernel shortcutting. Kernel shortcutting is the process of skipping command translations, meaning that rather than calling a syscall that must then be translated into the appropriate kernel function, the programmer can instead directly call the kernel function and "shortcut" the syscall path. This shortcutting can reap extreme benefits in terms of speed, especially in network applications where throughput and latency are of chief concern. I have seen and implemented the kernel shortcutting optimizations in a single threaded environment, but multithreaded environments offer an extremely tricky case, with the possibility of huge gains in overall program performance.

Application:

Multithreaded, shortcutted memcached could offer massive benefits in terms of throughput and latency, enabling a huge increase in performance for this network application. This is extremely useful for any online service that caters to large pools of clients, and implementing this capability would show to outside parties the viability of a chronokernel as a robust tool that could gain traction in the wider community.

Literature review:

https://www.kernel.org/doc/html/latest/trace/kprobes.html

KProbes offer much of the same capabilities as the chronokernel in terms of taking information from the kernel and its processes and revealing them to the user. The mechanism involves placing INT3 commands at places in the kernel, either at the beginning for kprobes or at the end for kretprobes, and then hijacking the interrupt handler to instead do what the kprobe is asking for, mainly pulling data from that point in the kernel. Earlier in the development of the chronokernel, and in shortcutting redis, the INT3 interrupt was used, in a much less elegant way, to achieve the same results. Overall, looking at kprobes allows the project to see what is possible at the end of development, and gives the project an ideal to strive toward.

https://ieeexplore.ieee.org/document/7280137

Here, researchers created a Linux kernel module, an outside piece of software with privileged access to kernel operations that runs as the kernel, that allows users to more finely and accurately

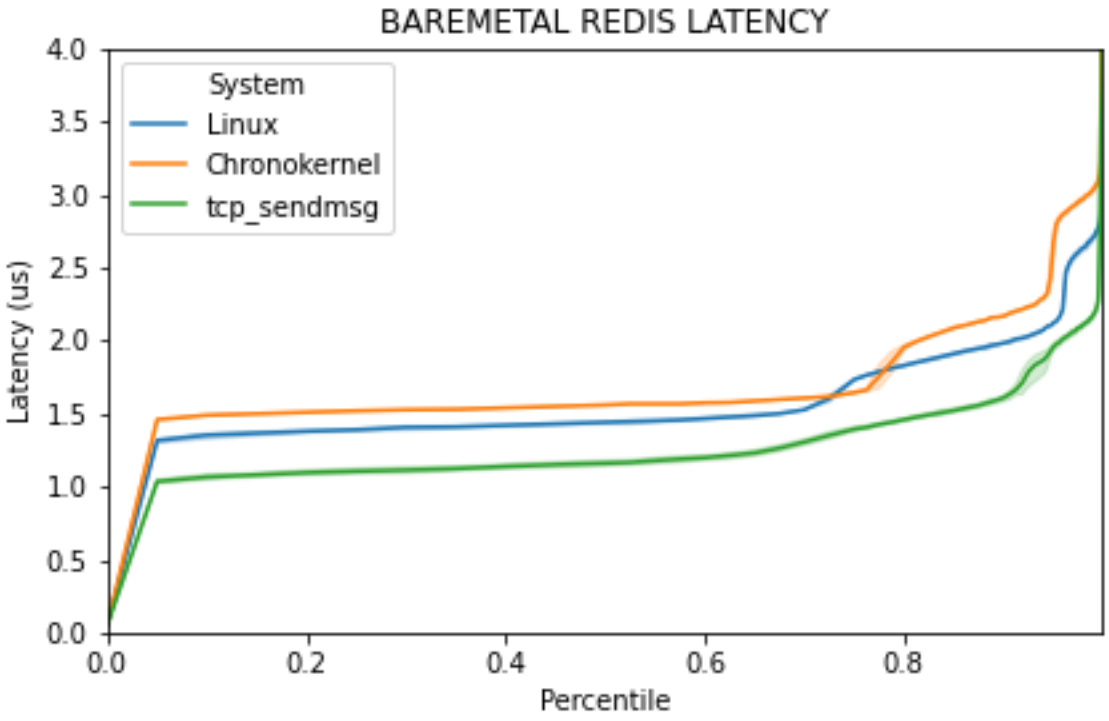
inspect processes being run by the host machine. This is achieved through separate kernel and user space programs, the user space program being extremely lightweight, that allow data to be transferred up and down from kernel to user space. This system shows disjointed similarities to the chronokernel in this project, with the difference being that the kernel and user space program in a chronokernel can be one and the same, with the user space program able to dynamically enter and exit kernel space as it sees fit.

https://ieeexplore.ieee.org/document/7082799?arnumber=7082799

Here, researchers attempted to do a hardware shortcut of memcached as opposed to a software shortcut as I am proposing. Their implementation utilized an FPGA attached to the host CPU's network interface card (NIC). When the NIC would receive a memcached request, the FPGA would attempt to satisfy the request, and if there was no way to do so, the request would be passed along to the host CPU and the normal memcached operation would occur. The researchers found that the FPGA was able to attain a miss rate of less than 30% utilizing an FPGA connected to a memory ½ that of the host CPU and using the Latest data distribution (the most recently accessed data is placed at the very front). Even though these numbers for the Latest data distribution were promising, the researchers concluded that a Zipf data distribution would still be more efficient at higher workloads. Overall, I wish that the researchers had been able to capture latency and throughput improvements with their FPGA shortcut implemented, as the miss rate only tells a fraction of the possible story.

Depth Exam (Thomas Unger)

Here, Thomas Unger argues that the current state of OS's needs to expand to allow for more flexibility for application developers. Currently, general purpose and special purpose OSes are two completely different categories, with general purpose OSes offering many advantages in terms of flexibility, but at the cost sacrificing optimizability; This tradeoff is reversed in the special purpose OS case. Thomas Unger argues that the ability to "elevate" and "lower" privilege level, from user to supervisor and back, effectively modifying the OS from general to special and back to general purpose on the fly, offers the best tradeoff between the two options. He calls this idea of a malleable OS the "chronokernel" and explains the many different abilities that this OS offers, of main focus to this project is application shortcutting. Below is a graph of the speedup experienced in single threaded redis utilizing application shortcutting in this chronokernel compared to baseline vanilla Linux. These results are what I hope to replicate with memcached.


BAREMETAL REDIS LATENCY
System
Linux
Chronokernel
tcp_sendmsg
Latency (us)
Percentile
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
0.0
0.2
0.4
0.6
0.8