Resourceful

Lucian Carata
James Snee
Oliver Chick
Ripduman Sohan
The Problem

- Easily understanding (kernel space) resource consumption for parts of an application
- Explaining variability in terms of resource consumption

```c
write(fd, buffer, BF_SZ) 1 us
...
write(fd, buffer, BF_SZ) 10 us
```
Resourceful

- Kernel module
  - Inserts kprobes in the kernel and does the resource accounting
  - Exposed as character device for mmap-ing

- API

  rscfl_acct(..., NEXT, group_ID);
  write(fd, buffer, BF_SZ);
  ...
  rscfl_read(..., &accounting);
Resourceful (the interesting bits)

- Kernel functional subsystem identification
  
  ```
  sys_accept (call graph)
  \ 
  … sock_alloc_file() ...
  … new_inode_pseudo() 
  ```

- Accounting for asynchronous resource consumption
Suggestion & Questions?

- getrusage
- ftrace
- iotop/netstat
- Perf
- Dtrace/SystemTap
- Magpie/Fay
- Dapper/X-Trace
- X-ray

Interested? Come talk to us: Lucian, James, Ollie, Rip

Lucian Carata
@lc525
Overheads

\[\Delta_{\text{median}} = 3.90\% \]
\[\Delta_{\text{average}} = 4.34\% \]
\[\Delta_{99^{\text{th}} \text{ percentile}} = 7.66\% \]
Resourceful

Lucian Carata
James Snee
Oliver Chick
Ripduman Sohan

- All team members part of Digital Technology Group
- Lucian Carata (presenting), supervised by Prof. Andy Hopper
The Problem

• Easily understanding (kernel space) resource consumption for **parts** of an application

• Explaining variability in terms of resource consumption

   ```
   write(fd, buffer, BF_SZ) 1 us
   ...
   write(fd, buffer, BF_SZ) 10 us
   ```

1. “parts” = function calls / application defined (i.e. all the syscalls made while servicing a user request)
2. The example of writes taking different times is simple to explain (one write was buffered). However, we aim to explain variability in terms of kernel subsystems for more complex scenarios (resources consumed by a user request; why was a request slower than another?)

3. (optional) Compared to ftrace, there is no “log processing” step, and we get more data besides time (i.e. nr. of TCP retransmissions, memory allocated/deallocated, cache misses)
Resourceful

- Kernel module
  - Inserts kprobes in the kernel and does the resource accounting
  - Exposed as character device for mmap-ing

- API
  
  ```
  rscfl_acct(..., NEXT, group_ID);
  write(fd, buffer, BF_SZ);
  ...
  rscfl_read(..., &accounting);
  ```

High level overview

1. Minimal amount of kprobes for breaking down accounting per kernel subsystem
   * each application thread gets its own mmap-ed memory for resources consumed within it.

2. group_ID (application level aggregators)

3. On read, the app “sees” the resource data in its own memory space
Resourceful (the interesting bits)

* Kernel functional subsystem identification

  ```
  sys_accept (call graph)
  \  
  ... sock_alloc_file()  NET
  ... new_inode_pseudo()  subsystem boundary (kprobe)
  VFS
  ```

* Accounting for asynchronous resource consumption

1. We identify kernel subsystems with Cscope (getting a kernel call graph) + directory structure for determining the minimal number of probes that need to be inserted

2. Asynchronous accounting: I/O Buffers (simple example)
Demo for Latency Explorer
Suggestions & Questions?

- getusage
- ftrace
- iotop/netstat
- Perf
- Dtrace/SystemTap
- Magpie/Fay
- Dapper/X-Trace
- X-ray

Lucian Carata
@lc525

Interested? Come talk to us:
Lucian, James, Ollie, Rip

Related systems on the left (for compare & contrast)
Two latency distributions, overlayed. The one below (in green) is lighttpd only, the one above (blue) lighttpd + resource accounting with resourceful. Median latency increases by 3.9%
Overall architecture of Resourceful