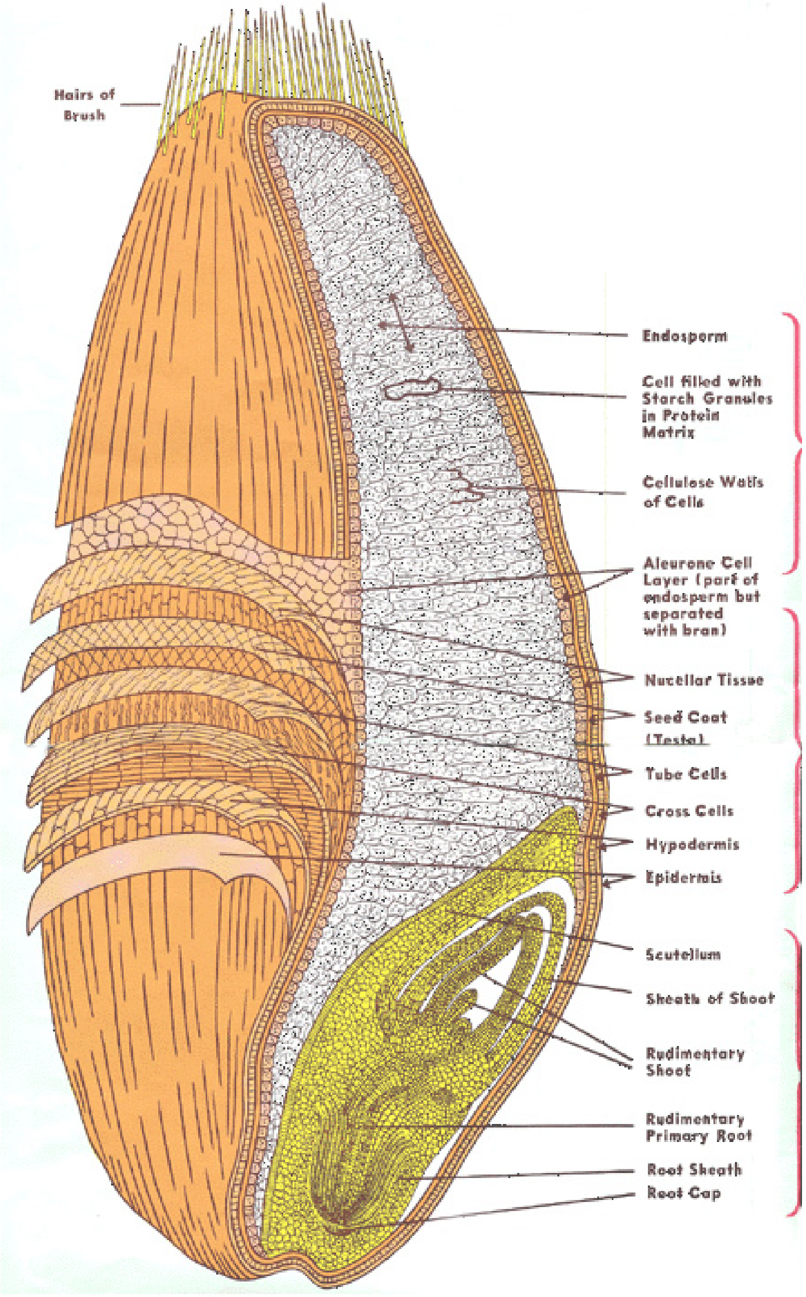
Resourceful

Lucian Carata
James Snee
Oliver Chick
Ripduman Sohan

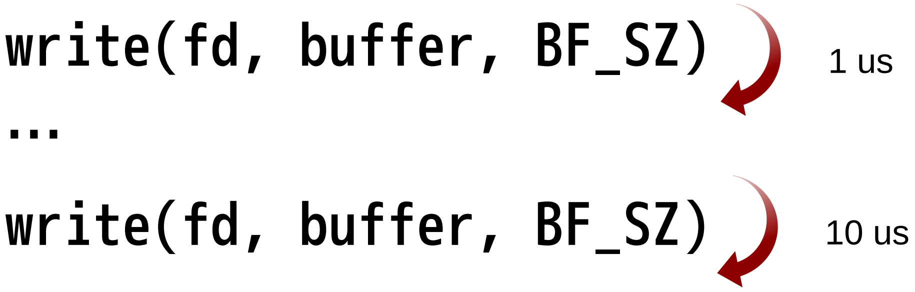


SRG/NetOS talklet, Sept 23rd 2014

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)
...
write(fd, buffer, BF_SZ)
```

1 us

10 us

Resourceful

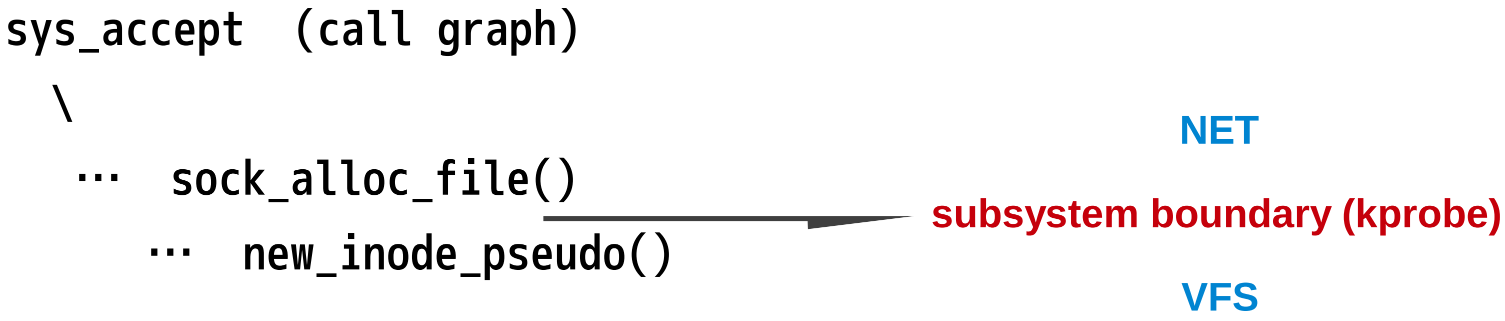
- Kernel module
  - Inserts kprobes in the kernel and does the resource accounting
  - Exposed as character device for mmap-ing

- 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()
```

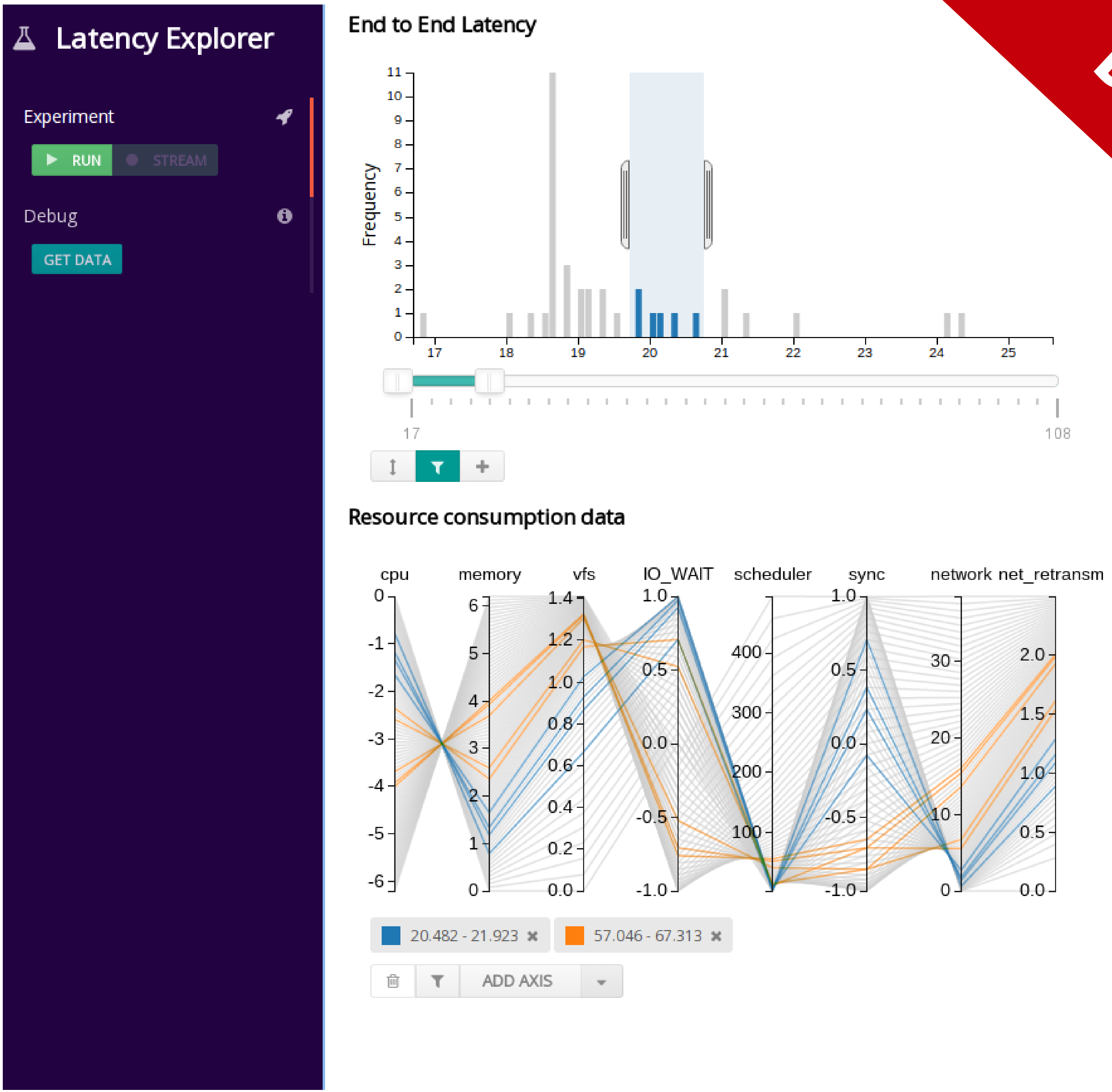
NET

subsystem boundary (kprobe)

VFS

- Accounting for asynchronous resource consumption

Demo Time !



Latency Explorer
Experiment
RUN
STREAM
Debug
GET DATA
End to End Latency
Frequency
0
1
2
3
4
5
6
7
8
9
10
11
17
18
19
20
21
22
23
24
25
17
108
Resource consumption data
cpu
memory
vfs
IO_WAIT
scheduler
sync
network
net_retransm
0
-1
-2
-3
-4
-5
-6
6
5
4
3
2
1
0
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.5
0.0
-0.5
-1.0
400
300
200
100
1.0
0.5
0.0
-0.5
-1.0
30
20
10
0
2.0
1.5
1.0
0.5
0.0
🟦 20.482 - 21.923 ✖
🟧 57.046 - 67.313 ✖
ADD AXIS

Suggestions & Questions ?

getrusage
ftrace
iotop/netstat
Perf

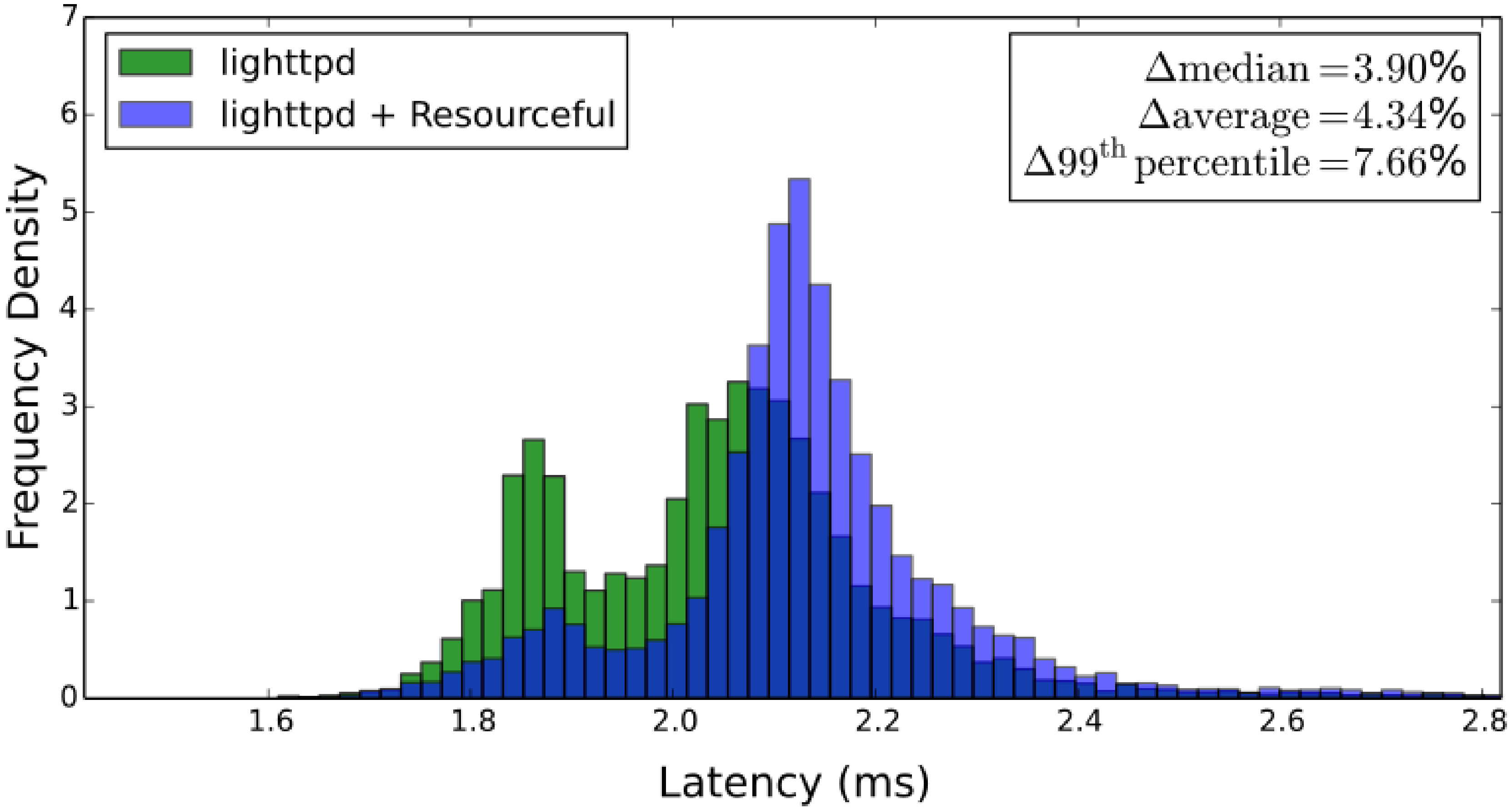
Dtrace/SystemTap

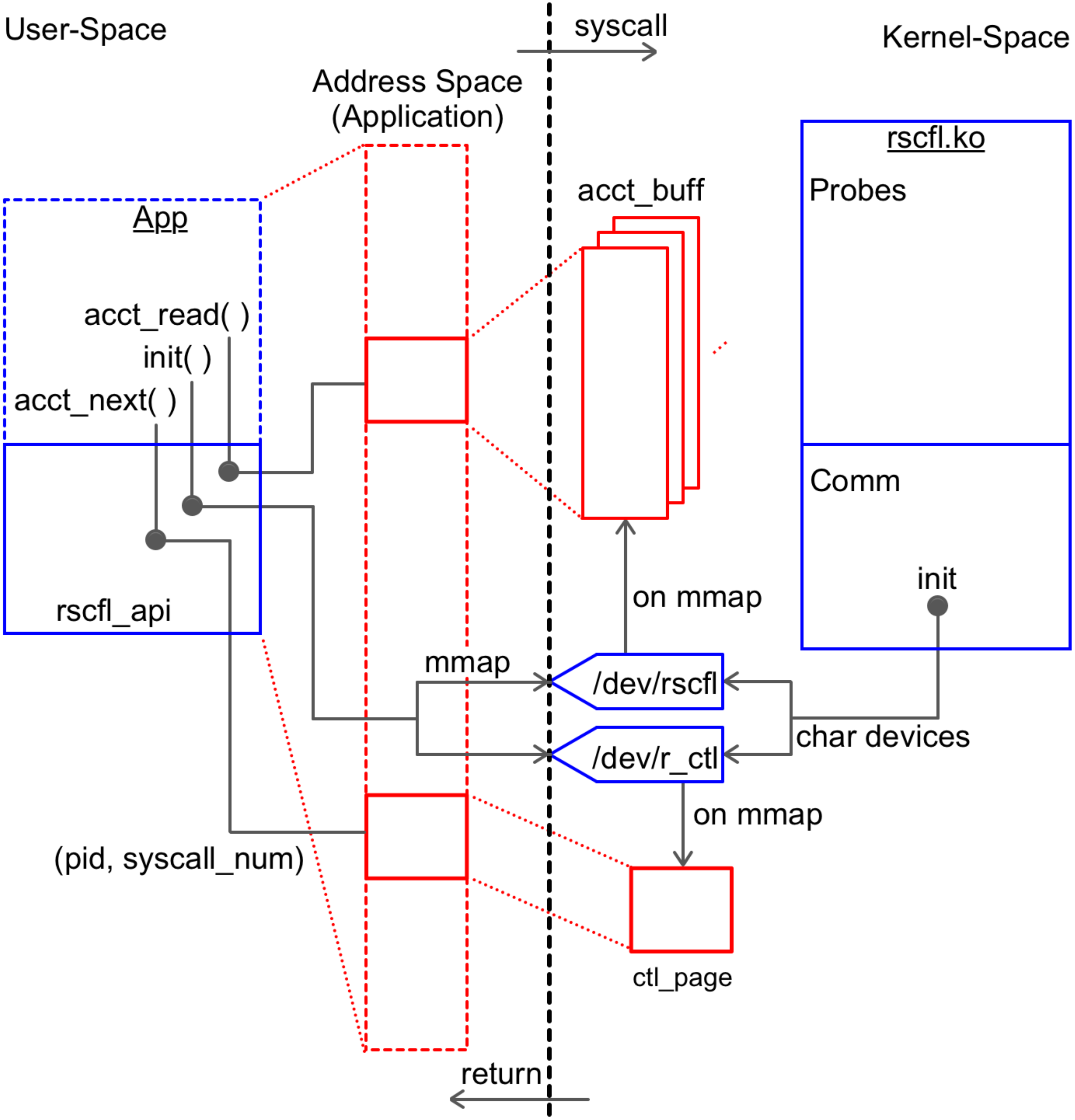
Magpie/Fay
Dapper/X-Trace
X-ray

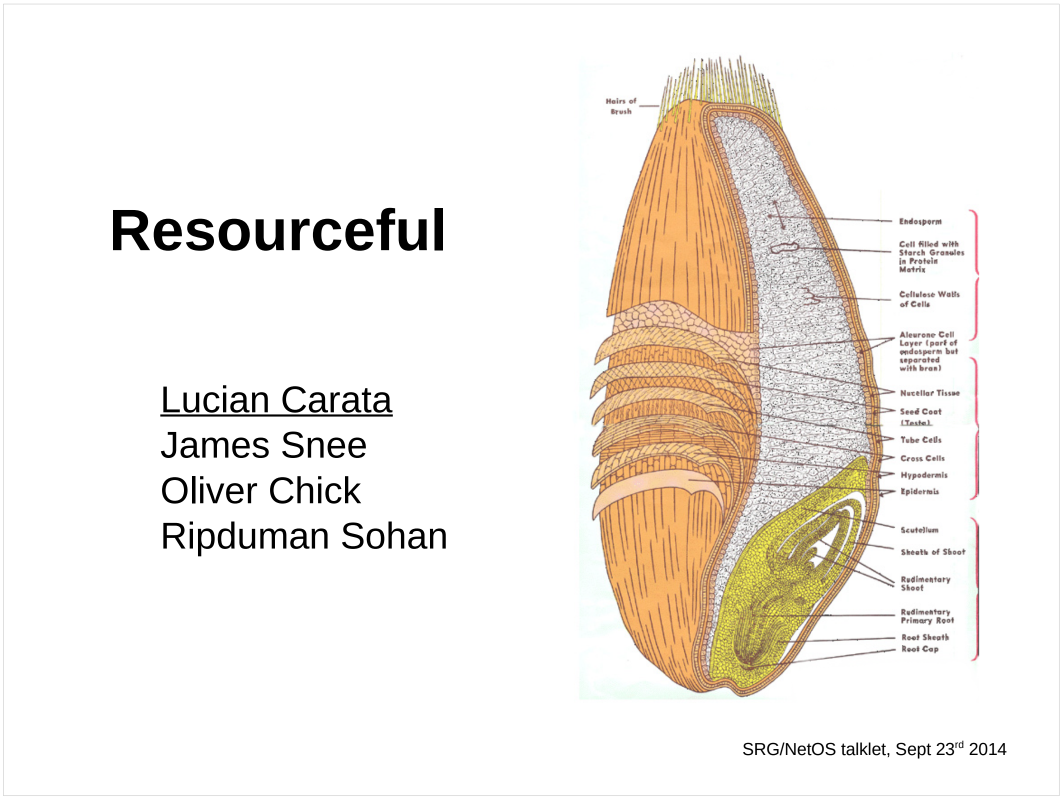
Lucian Carata
@**lc525**

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

Overheads


lighttpd
lighttpd + Resourceful
Δmedian = 3.90%
Δaverage = 4.34%
$\Delta 99^{\mathrm{th}}$ percentile = 7.66%
Frequency Density
0
1
2
3
4
5
6
7
1.6
1.8
2.0
2.2
2.4
2.6
2.8
Latency (ms)


User-Space
syscall
Kernel-Space
Address Space
(Application)
rscfl.ko
Probes
App
acct_buff
acct_read( )
init( )
acct_next( )
Comm
rscfl_api
init
on mmap
mmap
/dev/rscfl
char devices
/dev/r_ctl
on mmap
(pid, syscall_num)
ctl_page
return



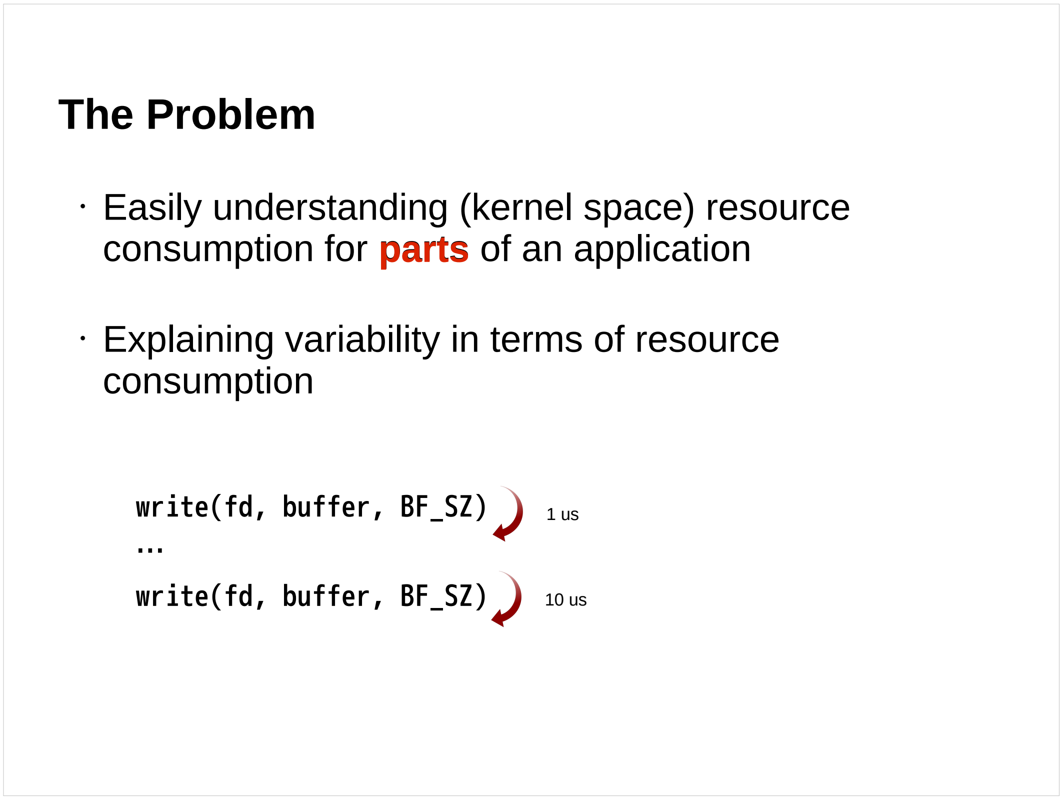
Resourceful

<u>Lucian Carata</u>
James Snee
Oliver Chick
Ripduman Sohan

Hairs of Brush
Endosperm
Cell filled with Starch Granules in Protein Matrix
Cellulose Walls of Cells
Aleurone Cell Layer (part of endosperm but separated with bran)
Nucellar Tissue
Seed Coat (Testa)
Tube Cells
Cross Cells
Hypodermis
Epidermis
Scutellum
Sheath of Shoot
Rudimentary Shoot
Rudimentary Primary Root
Root Sheath
Root Cap

SRG/NetOS talklet, Sept 23rd 2014

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



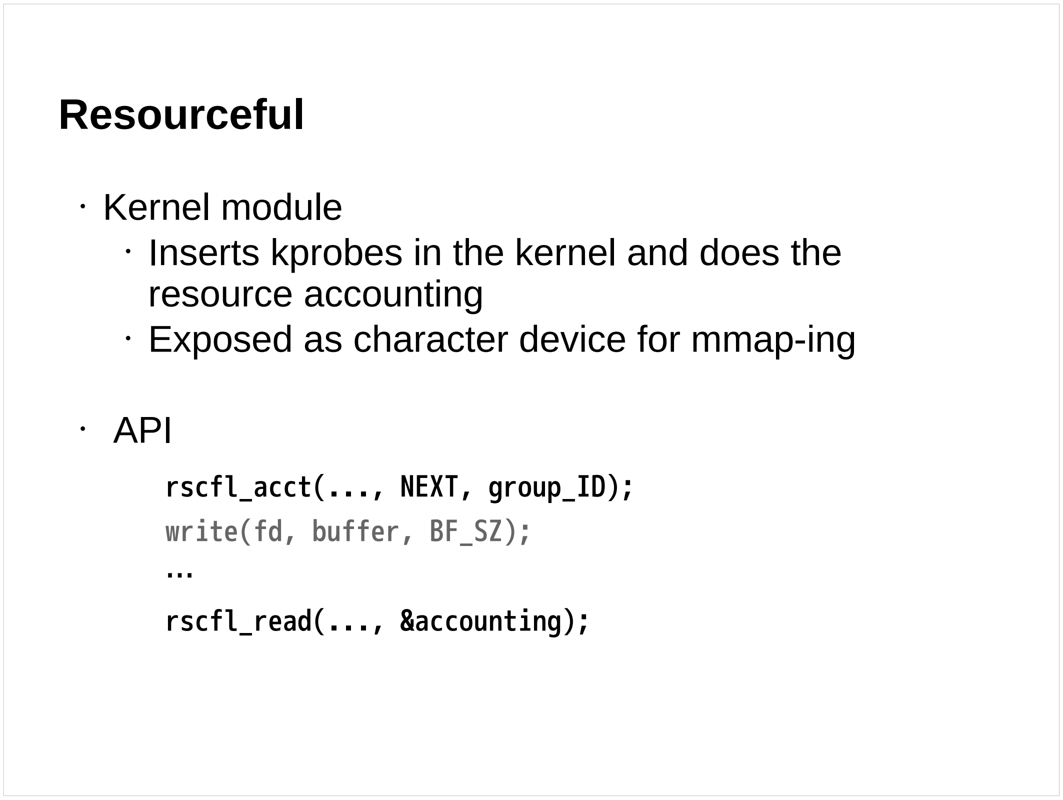
The Problem

- 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 (ie 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

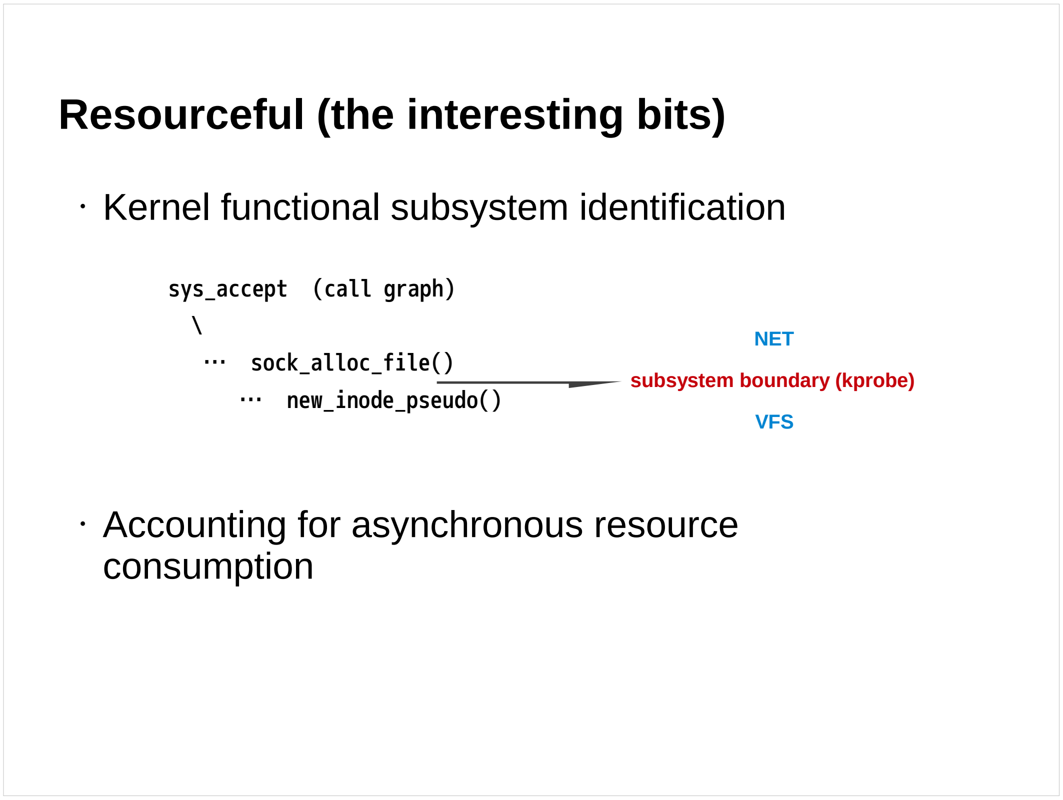
```
rscfl_acct(..., NEXT, group_ID);
write(fd, buffer, BF_SZ);
...
rscfl_read(..., &accounting);
```

High level overview

1. Minimal ammount of kprobes for breaking down accounting per kernel subsystem
   * each application thread gets it's 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()
      ···  new_inode_pseudo()
```

NET

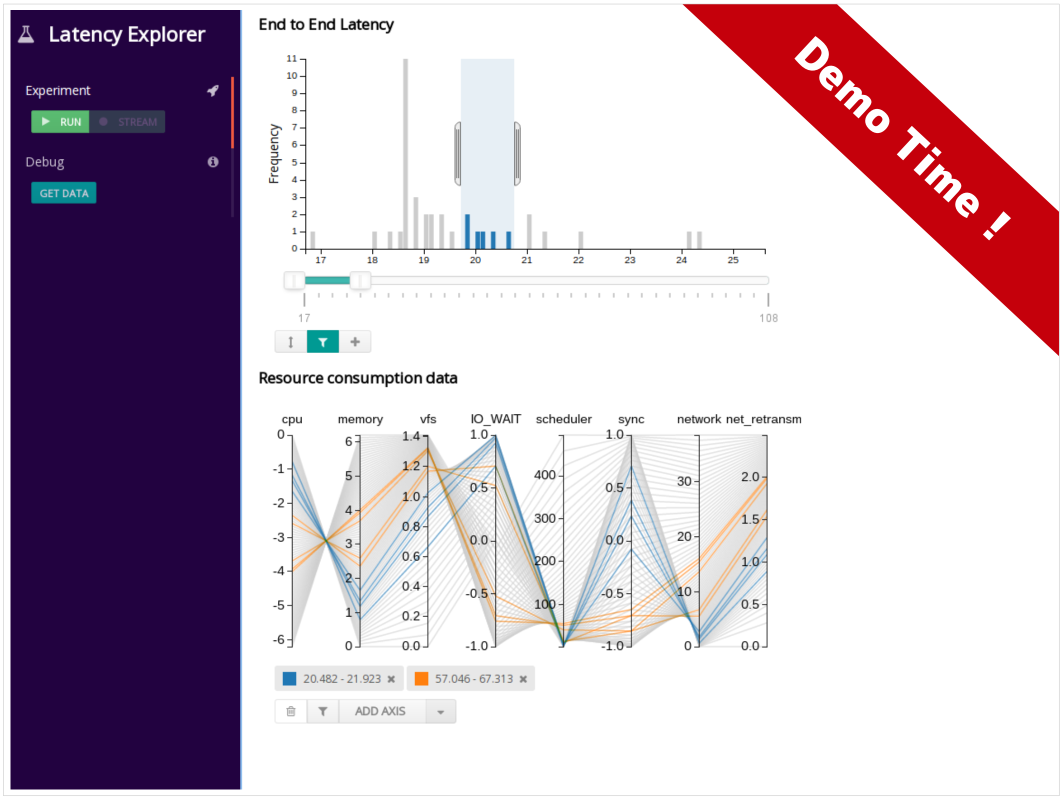
subsystem boundary (kprobe)

VFS

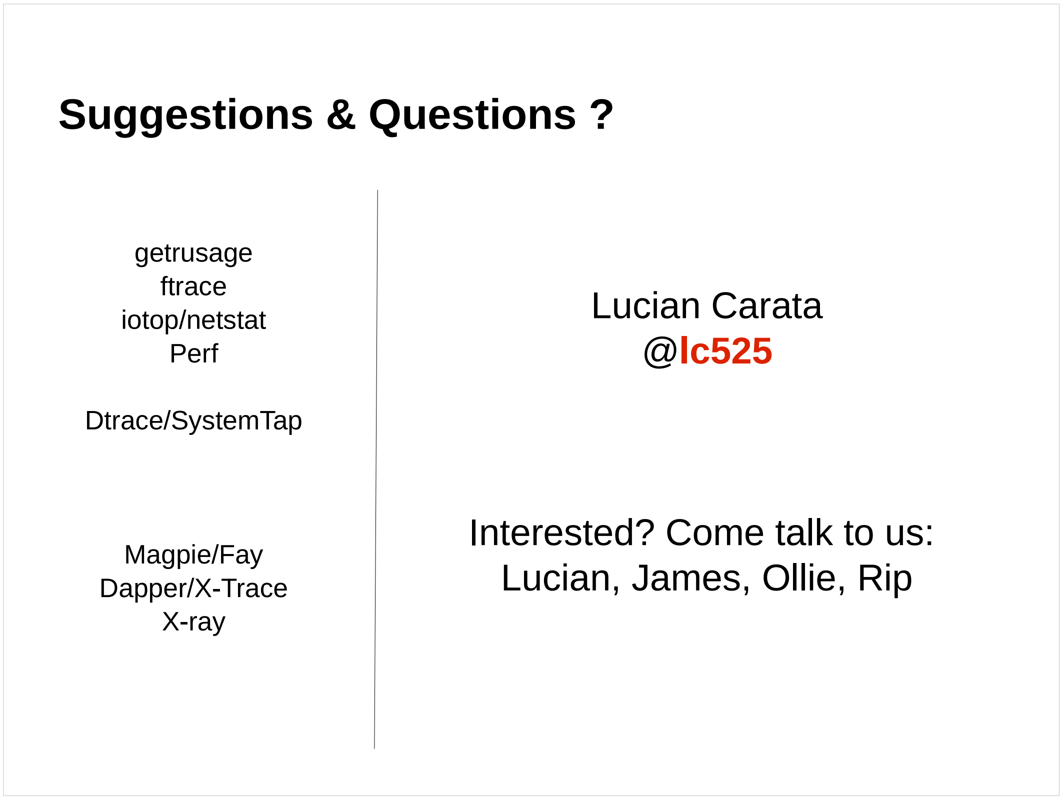
- Accounting for asynchronous resource consumption

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 ?

getrusage
ftrace
iotop/netstat
Perf

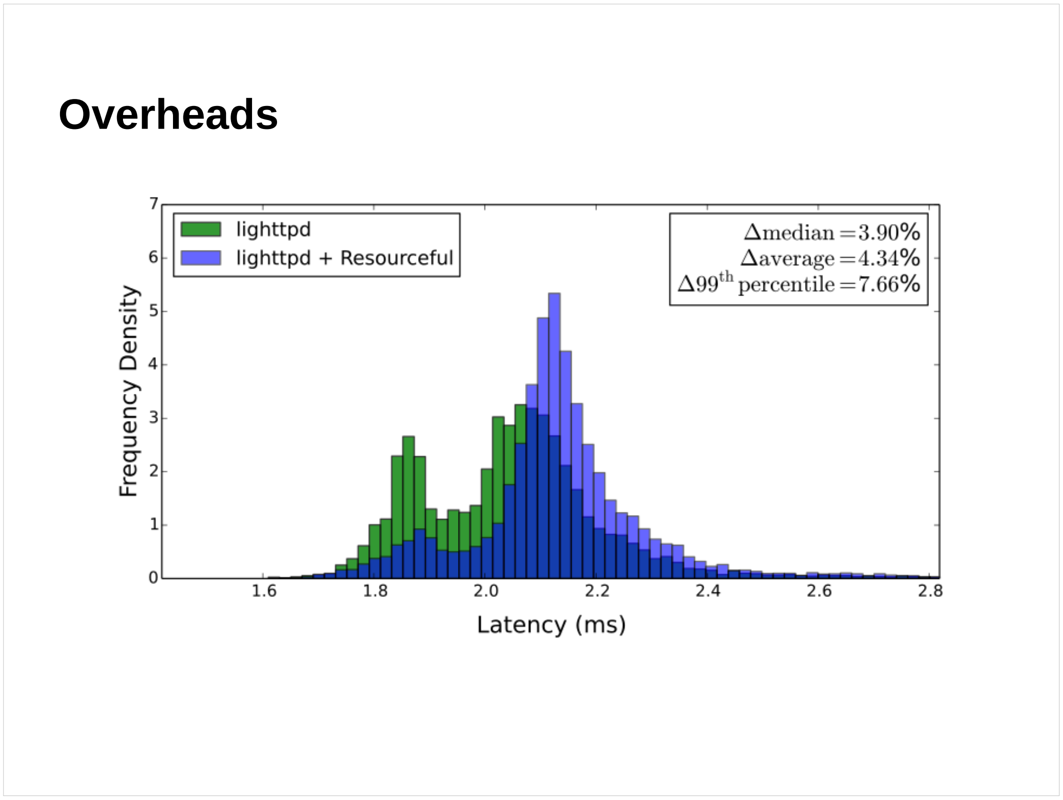
Dtrace/SystemTap

Magpie/Fay
Dapper/X-Trace
X-ray

Lucian Carata
@lc525

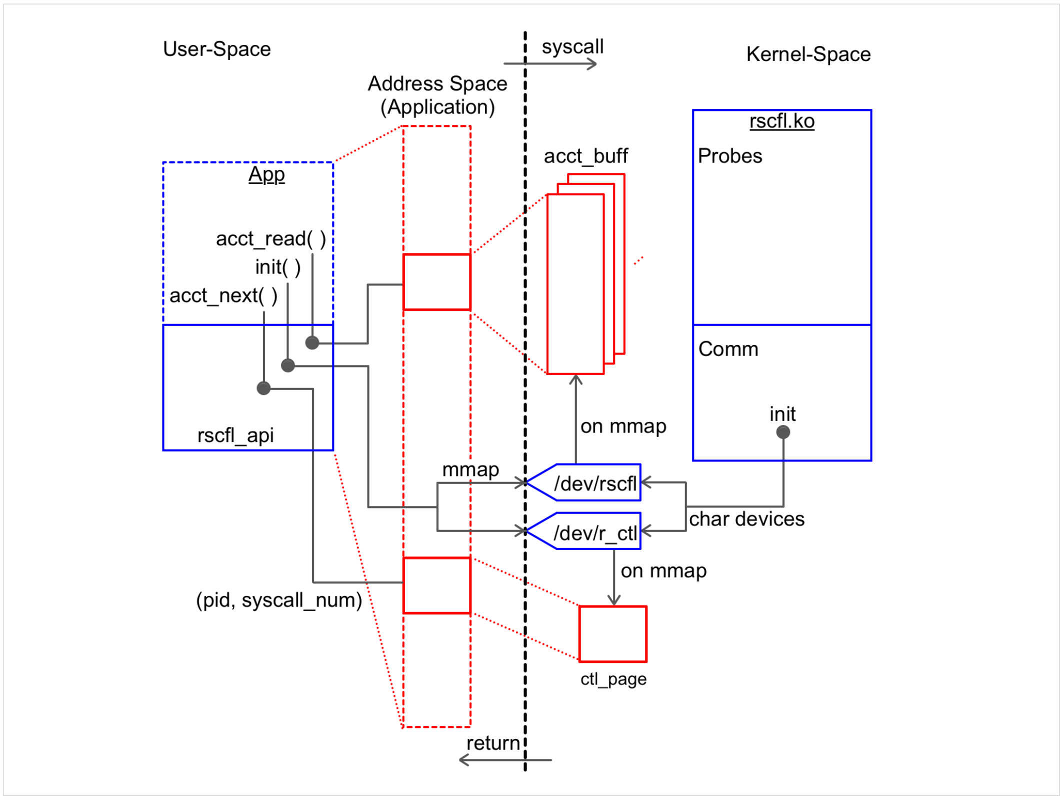
Interested? Come talk to us:
Lucian, James, Ollie, Rip

Related systems on the left (for compare & contrast)



Overheads

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