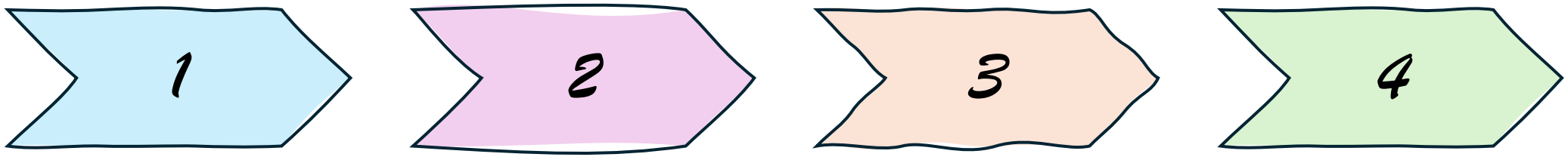
Data-Aware Adaptive Tracing

Ali Entezari

PRM - May 2025

Outline



Motivation Methodology Tools Objectives

Motivation

ASPECT	Adaptive Data-Aware Tracing	Full-Fidelity Tracing
Overhead Reduction	✓	✗
Data Load Management	✓	✗
Analysis Clarity	✓	✗
Scalability	✓	✗
Resource Efficiency	✓	✗

Methodology

Selective Tracing Triggered by Predefined, Unexpected or Abnormal Data

Idea 1: Representative Inputs of Equivalent Partitions

- Middle values
- Boundary values

Idea 2: Uncommon Function Inputs

- distribution of inputs by sampling (fixed/dynamic)

Methodology

Selective Tracing Triggered by Predefine
or Abnormal Data

```
int foo (int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

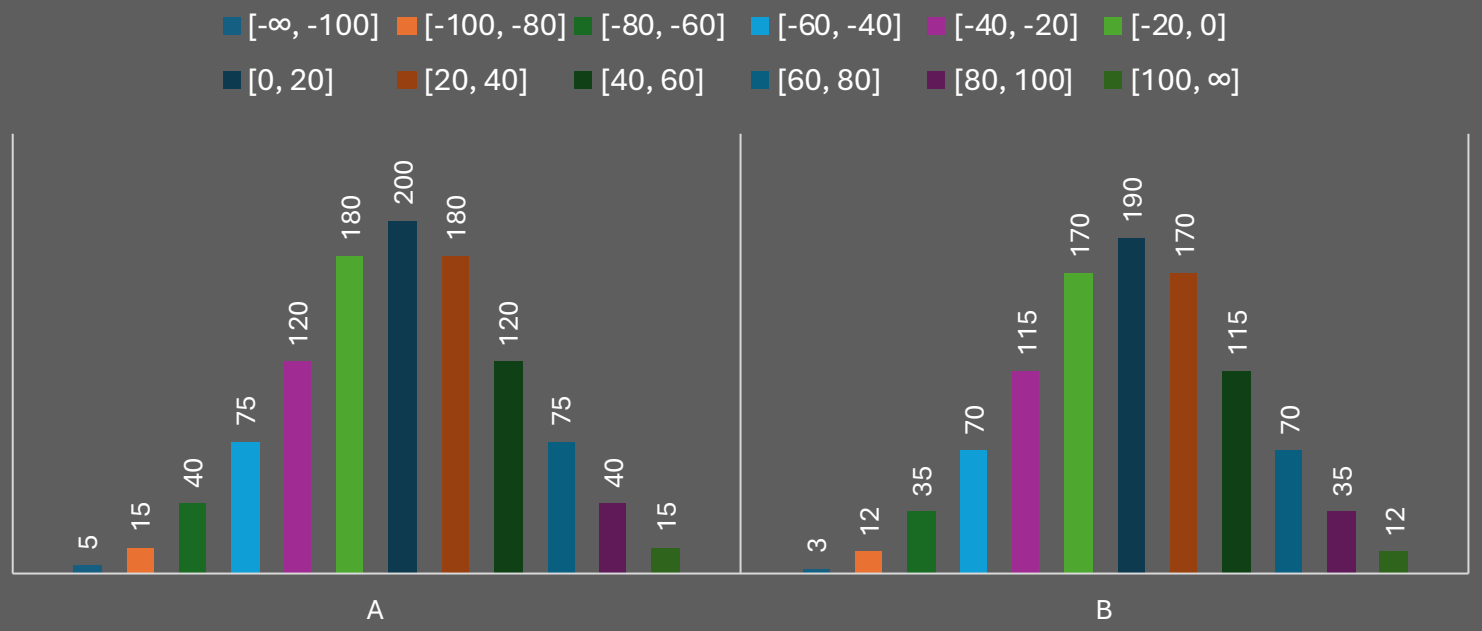
Idea 1: Representative Inputs of Equivalent Partitions

- Middle values
- Boundary values

range a: [-100, 100]

if a <= -100 or a >= 100 then trace on

INPUT VALUE REPETITION DISTRIBUTION




```
int foo (int a, int b)
{
    int sum;
    sum = a + b;
    return sum;
}
```

fine

rtitions

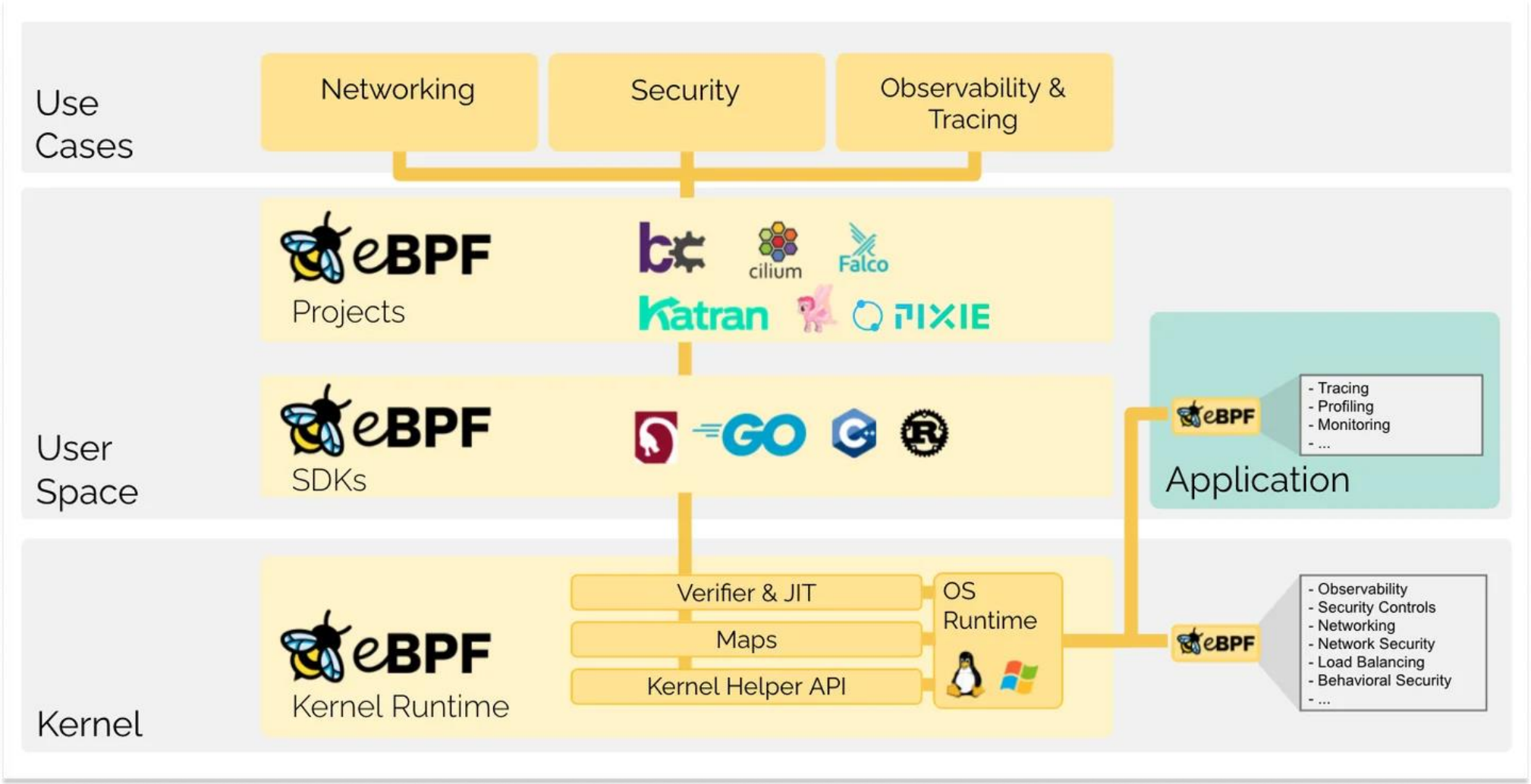
- Middle values
- Boundary values

R(a): repetition a
if R(a) < 10 then trace on

Idea 2: Uncommon Function Inputs

- distribution of inputs by sampling (fixed/dynamic)

eBPF


Use Cases
Networking
Security
Observability & Tracing
eBPF
Projects
bcc
cilium
Falco
Katran
PIXIE
User Space
eBPF
SDKs
GO
Kernel
eBPF
Kernel Runtime
Verifier & JIT
Maps
Kernel Helper API
OS Runtime
eBPF
Application
- Tracing
- Profiling
- Monitoring
- ...
eBPF
- Observability
- Security Controls
- Networking
- Network Security
- Load Balancing
- Behavioral Security
- ...

eBPF related tools

- **bpftrace:**
  High-level tracing language for Linux

- **bpftime:**
  Userspace eBPF runtime for Observability, Network & General extensions Framework

bpftrace example 1: calculate_sum

- Example summation function
- Attach uprobe to calculate_sum function
- Record ustack when a > 10 and b < 200

```
#!/usr/bin/env bpftrace

uprobe:/home/alient/user_apps/bin/example_program:calculate_sum
{
  printf("%s %d %d\n", comm, arg0, arg1);
  if (arg0 > 10 && arg1 < 200 )
  {
    @sus[comm, arg0, arg1]=ustack;
  }
}
```

bpftrace script (calsum.bt)

```
alient@Desltop:~/Codes/bpftrace_scripts$ sudo bpftrace calsum.bt
Attaching 1 probe...
example_program 5  300
example_program 42 100
example_program 50 500
^C

@sus[example_program, 42, 100]:
    calculate_sum+0
    __libc_start_call_main+122
    __libc_start_main@GLIBC_2.2.5+139
    _start+37
```

Recorded ustack

Terminal output

bpftrace example 2: ioctl

- Example ioctl bug*
- Attach uprobe to ioctl
- Record ustack when arg2 == 0x541c

```c
// author by ziiiro@thu
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>


int main(int argc, char** argv)
{
        int fd = open("/dev/tty1", O_RDWR, 0);
        unsigned short size1[3] = {3, 0x21, 0};
        ioctl(fd, 0x5609, size1); // VT_RESIZE
        for (int i = 0; i < 30; i++) {
            write(fd, "\x0a", 1);
        }

        signed int args[3] = {13, -5, 0};
        ioctl(fd, 0x541c, args); // TIOCLINUX
        unsigned short size2[3] = {3, 0x39, 0};
        ioctl(fd, 0x5609, size2); // VT_RESIZE
}
```

ioctl_example.c

```
alient@Desltop:~/Codes/bpftrace_scripts$ sudo bpftrace ioctl.bt
Attaching 1 probe...
Comm: ioctl_example, PID: 219954, FD: -1, Command: 0x541c, Arg: 0x7ffef36bf5dc
^C

@sus[ioctl_example]:
    0x7e17ded24db0
    0x7e17dec2a1ca
    0x7e17dec2a28b
    0x631ee0013025
```

Terminal output

```
#!/usr/bin/env bpftrace

uprobe:/lib/x86_64-linux-gnu/libc.so.6:ioctl
/ arg2 == 0x541c / {
  printf("Comm: %s, PID: %d, FD: %d, Command: 0x%x, Arg: 0x%lx\n", comm, pid, arg1, arg2, arg3);
  // Save the kernel stack trace in the map
  @sus [comm] = ustack;
}
```

bpftrace script (ioctl.bt)

* The example program bug has not been reproduced

Other tools (to be checked)

- uftrace
  https://github.com/namhyung/uftrace

- LTTng
  https://lttng.org/

Objectives

- Reduce **resource overhead** and the **performance overhead** compared to full tracing

- Maximize the number of recorded traces that lead to bugs

Next steps...

- Find inputs in open-source benchmarks that trigger **rare** or **erroneous** or **exceptional** branches

- Fuzzing/Random testing on Phoronix Test Suite OpenSSL unit tests

Thank you

Email: ali.entezari@polymtl.ca