A tale of two eBPFs: Towards Managing eBPF Programs for the Rest of US

Theophilus A. Benson
Carnegie Mellon University

About Me…

Professor @ ECE--CMU

- Addressing digital divide in global south
- Management frameworks for the cloud
- Web performance

CDN Performance

CDN Availability (ZDT)

Protocol Standardization

Cloud/Service Management

Network Management

Facebook, Google, Isovalent, Microsoft, and Netflix announce eBPF Foundation

Aug 12, 2021

Cloudflare architecture and how BPF eats the world

05/18/2019

Marek Maj

Recently at Ne

talk titled "Lin

matter the qu

Here is a trans

InfoQ

SIGN UP / LOGIN

June 13-15, 2023 **QCon New York**
Find real-world practical inspiration from the world's most innovative software leaders. Attend in-person.

Oct 2-6, 2023 **QCon San Francisco**
Learn what's next in software from world-class leaders pushing the boundaries. Attend in-person or online.

The Software Architec
Your monthly guide to all th technologies and technique professional needs to know free.

InfoQ Homepage > Articles > The Silent Platform Revolution: How EBPF Is Fundamentally Transforming Cloud-Native Platforms

DEVOPS

The Silent Platform Revolution: How eBPF Is Fundamentally Transforming Cloud-Native Platforms

Security/Monitoring (Emerging)
Falco, Tetragon, Pixie, **DDoS**: Cloudflare, FW: Cloudflare-Rakelimt,

Observability/Monitoring
Falco, Hubble, Sysdig, Tracee, RH OCP(Netobserv), Flowmill (not open-source), SkyDive, DataDog, Elastic

Networking:
FB-Katran, Cloudflare, Cilium, Isovalent-Cilium, Calico, Cloudflare-Rakelimt, Huawei-Mizar, K8s-envoy,

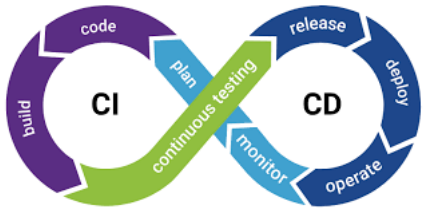
Popular Use cases Bias Our Views

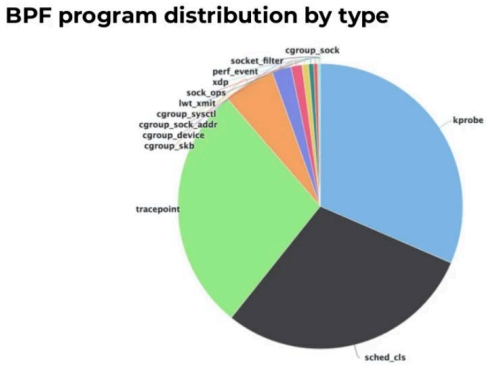
Stable Code Bases

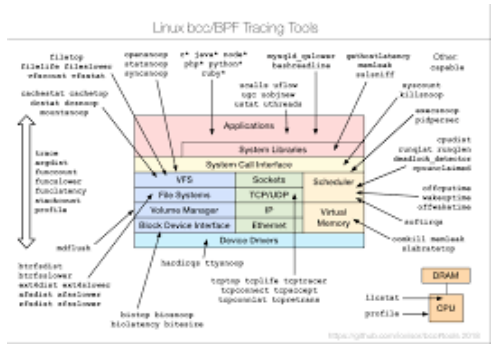
Small number of same Programs Deployed Everywhere

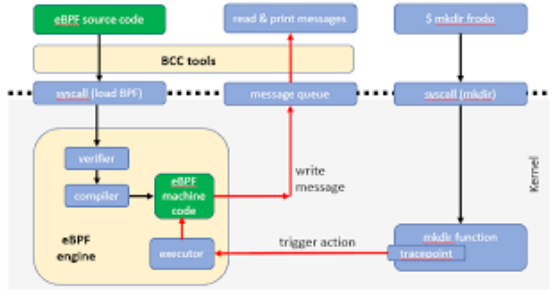
Generally Understood/trusted Code

No need to Write/Extend code









Testing Frameworks

Orchestration

Understand code

Extend/Edit Code

Debugging

Policy Enforcement

Ecosystem of DevTools to enable safe and quick incorporation/extension of third party eBPF code at scale

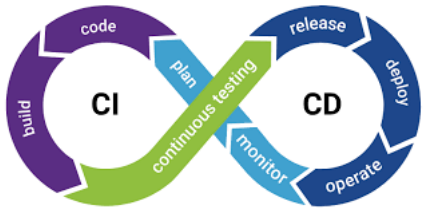
Popular Use cases Bias Our Views

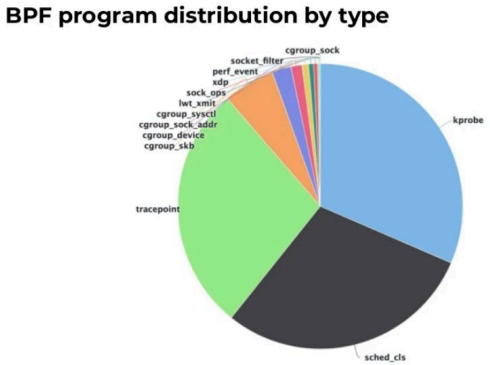
Stable Code Bases

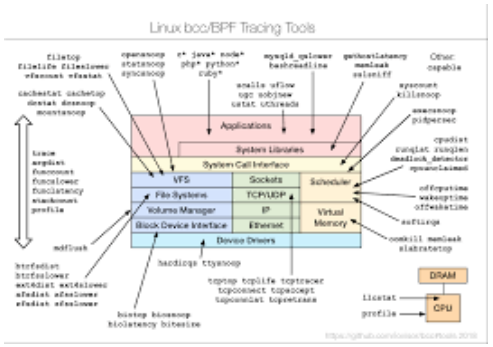
Small number of same Programs Deployed Everywhere

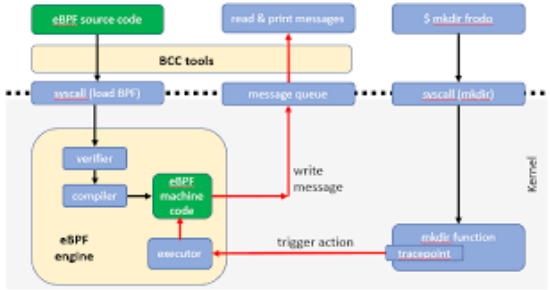
Generally Understood/trusted Code

No need to Write/Extend code









Testing Frameworks

Orchestration

Understand code

Extend/Edit Code

Debugging

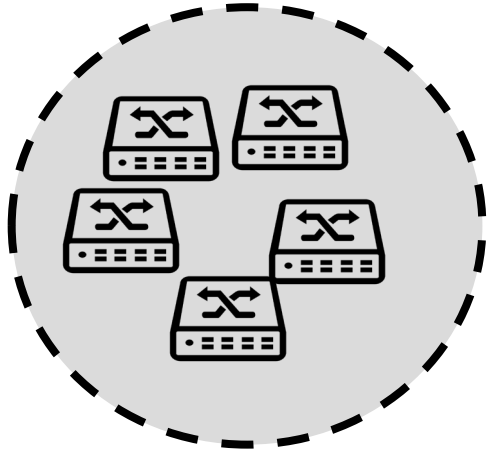
Policy Enforcement

Roadmap

Background

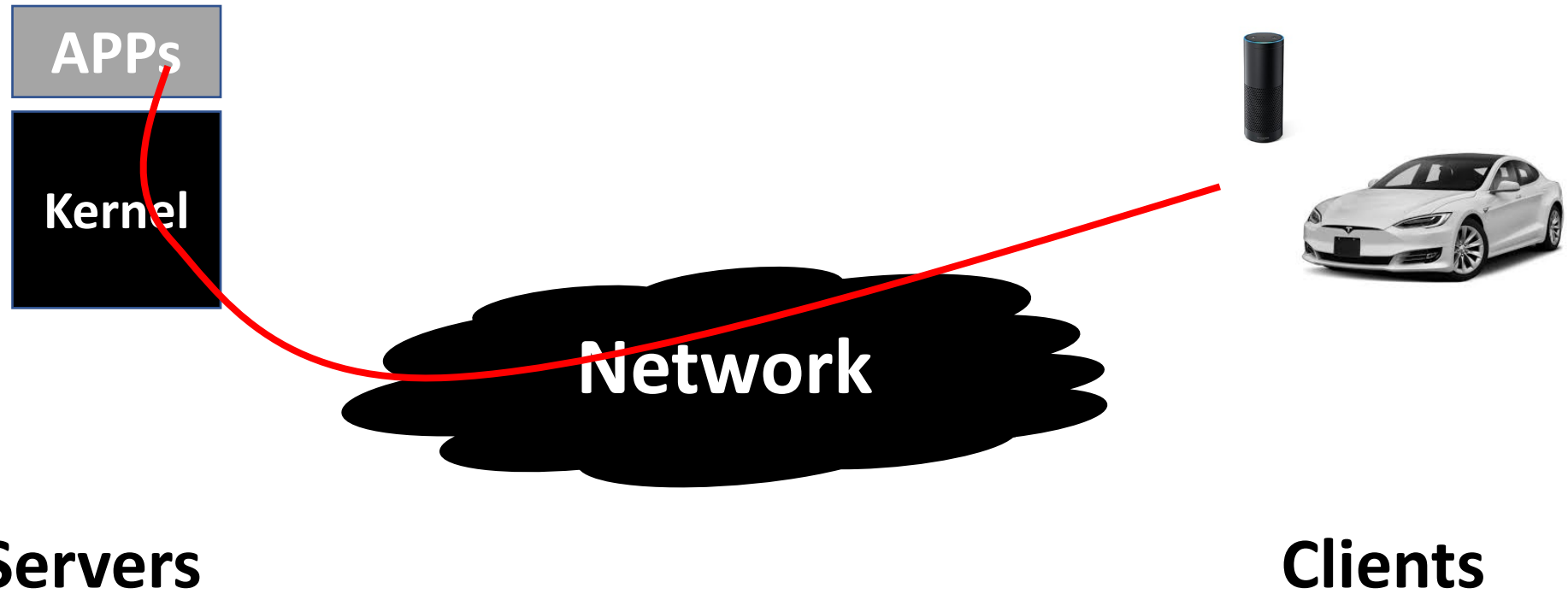
Understanding Code (Registry)

Extending Code (OPENED)

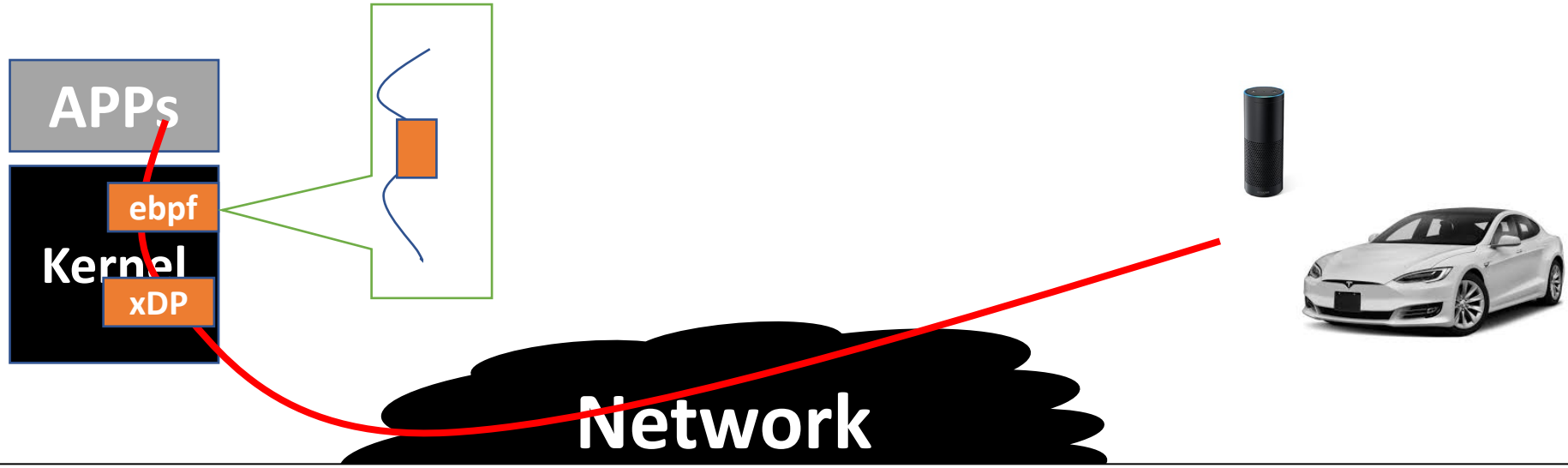


Orchestrating @Scale (Apiary)

Application Deployments – 10 Years ago.



Application Deployments – 10 Years ago.

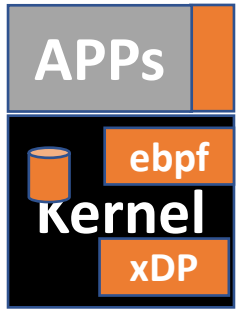


BPF: A New Type of Software

02 Dec 2019

At Netflix we have 15 BPF programs running on cloud servers by default; Facebook has 40. These programs are not processes or kernel modules, and don't appear in traditional observability tools. They are a new type of software, and make a fundamental change to a 50-year old kernel model by introducing a new interface for applications to make kernel requests, alongside syscalls.

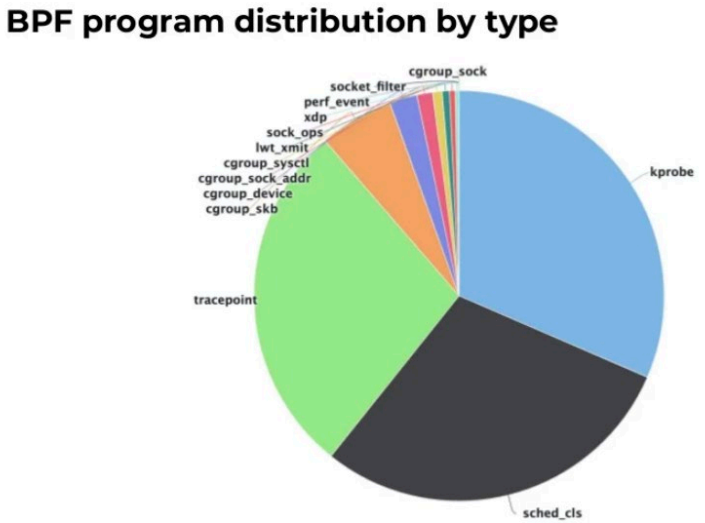
Key eBPF Primitives and Abstractions



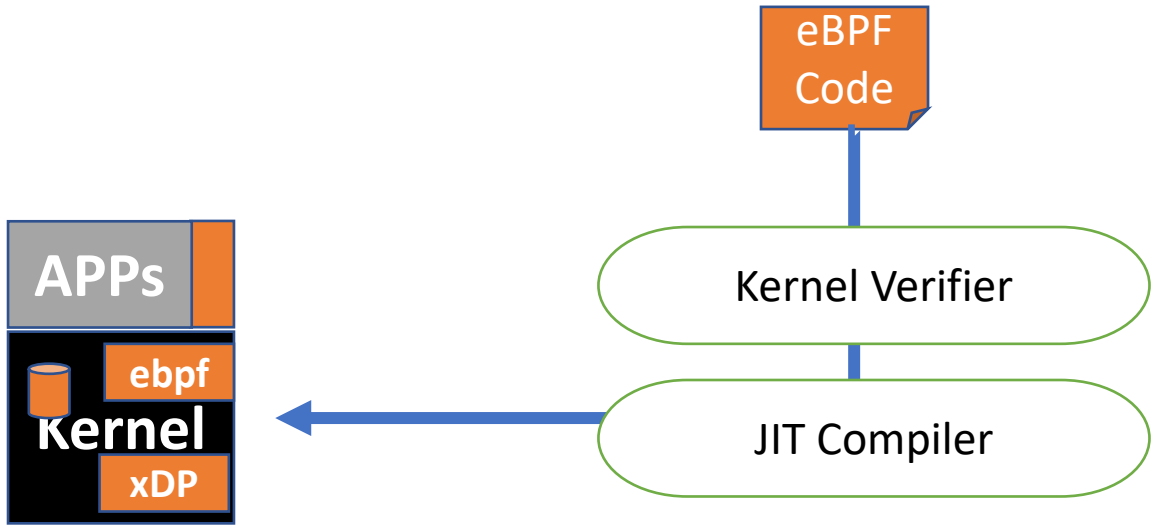
Kernel: eBPF Program
User space: Control

eBPF Program: Code pair + data structures

Hookpoints: where to attach eBPF programs



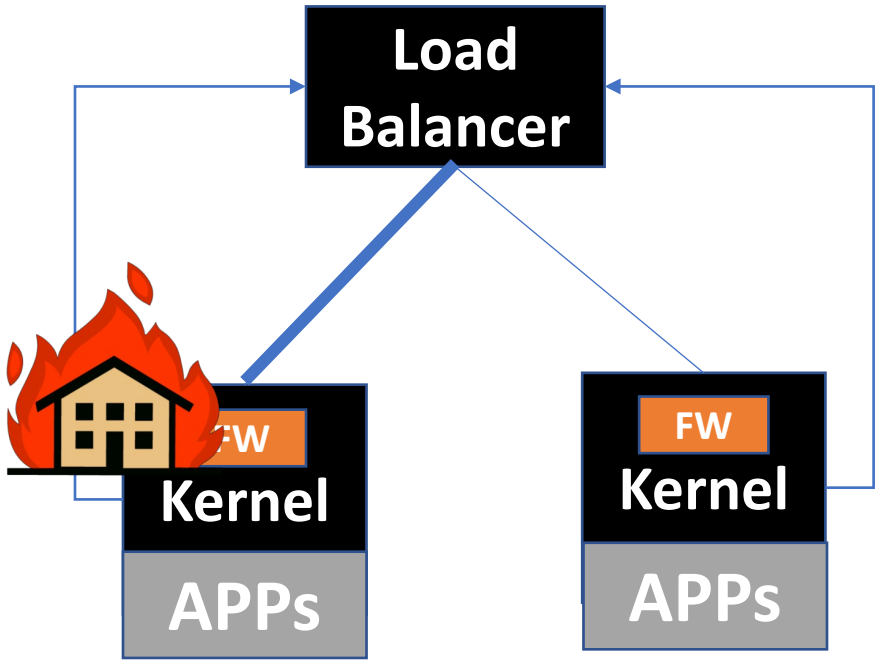
Key eBPF Primitives and Abstractions



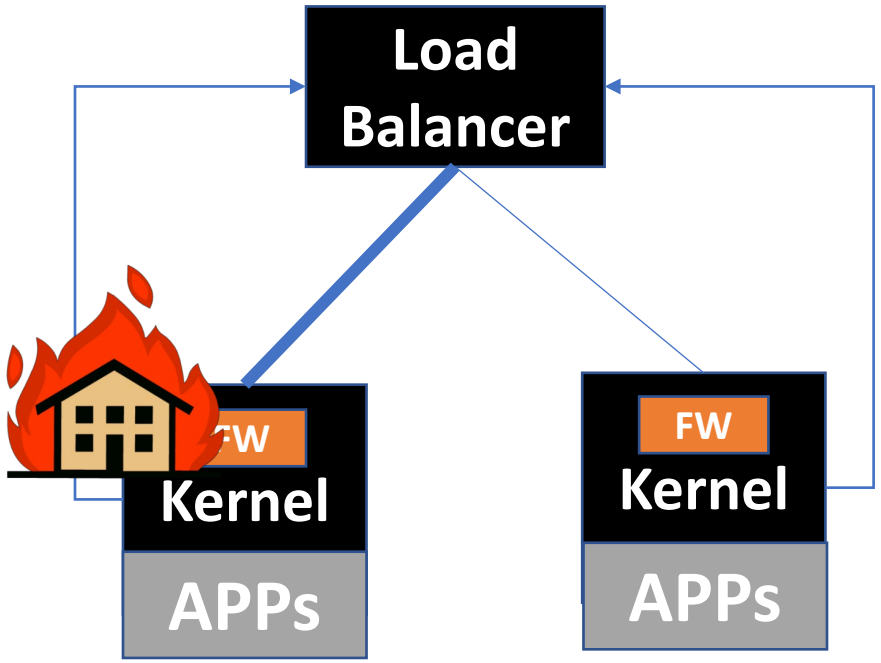
BPF: A New Type of Software

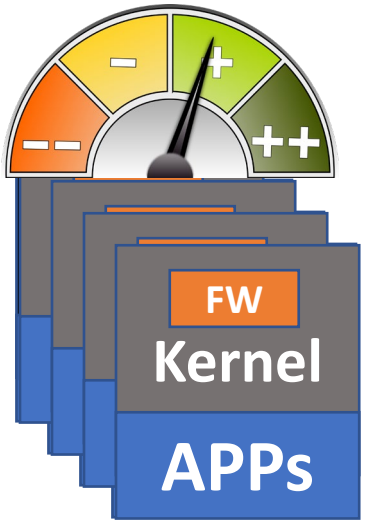
02 Dec 2019

At Netflix we have 15 BPF programs running on cloud servers by default; Facebook has 40. These programs are not processes or kernel modules, and don't appear in traditional observability tools. They are a new type of software, and make a fundamental change to a 50-year old kernel model by introducing a new interface for applications to make kernel requests, alongside syscalls.



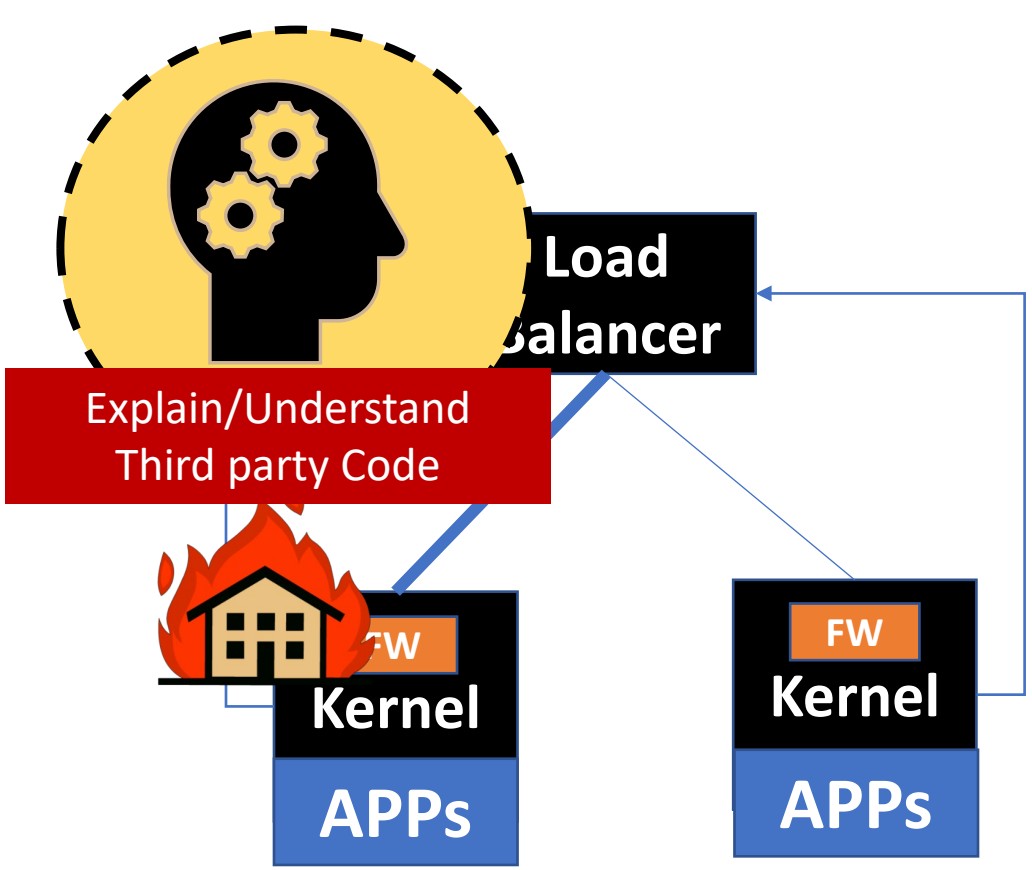
Introduce Mismatch between Application and Reality

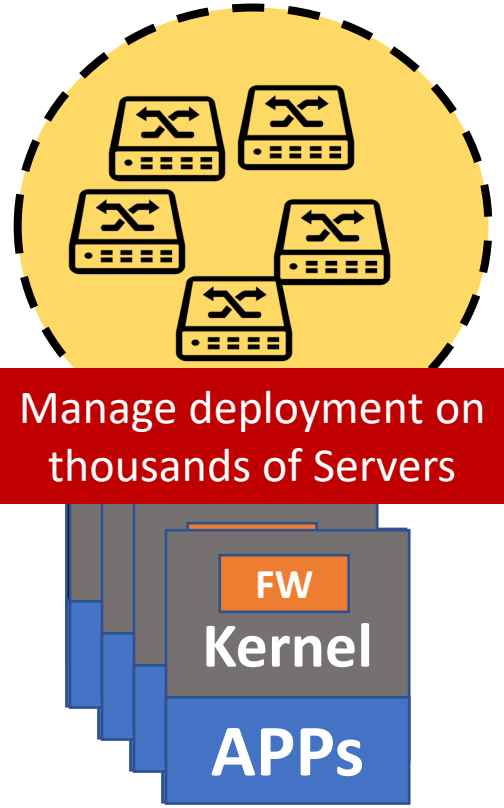




Introduce Mismatch between Application and Reality

Introduces unexpected and unpredictable performance variability

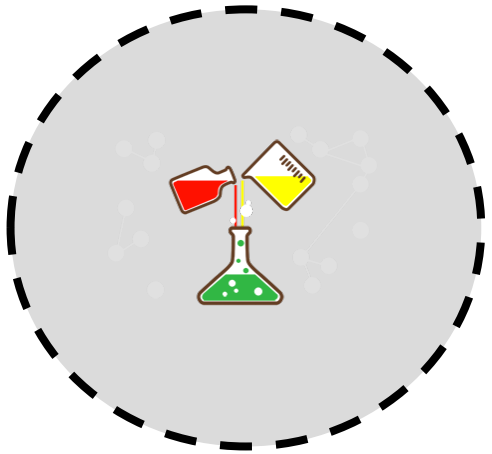




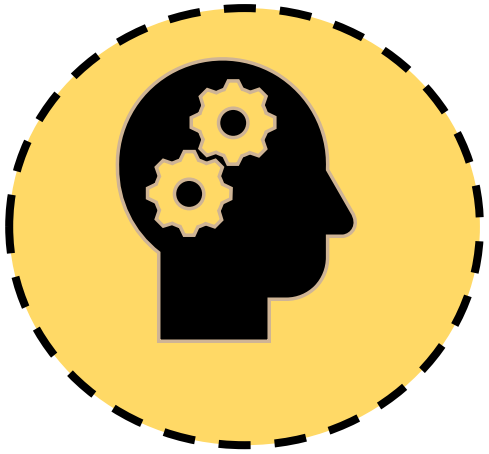
Introduce Mismatch between Application and Reality

Introduces unexpected and unpredictable performance variability

Roadmap

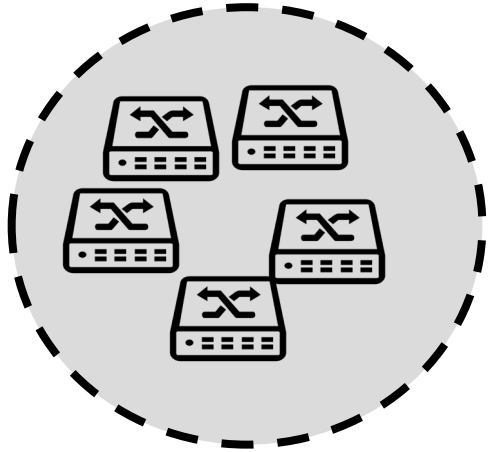


Background

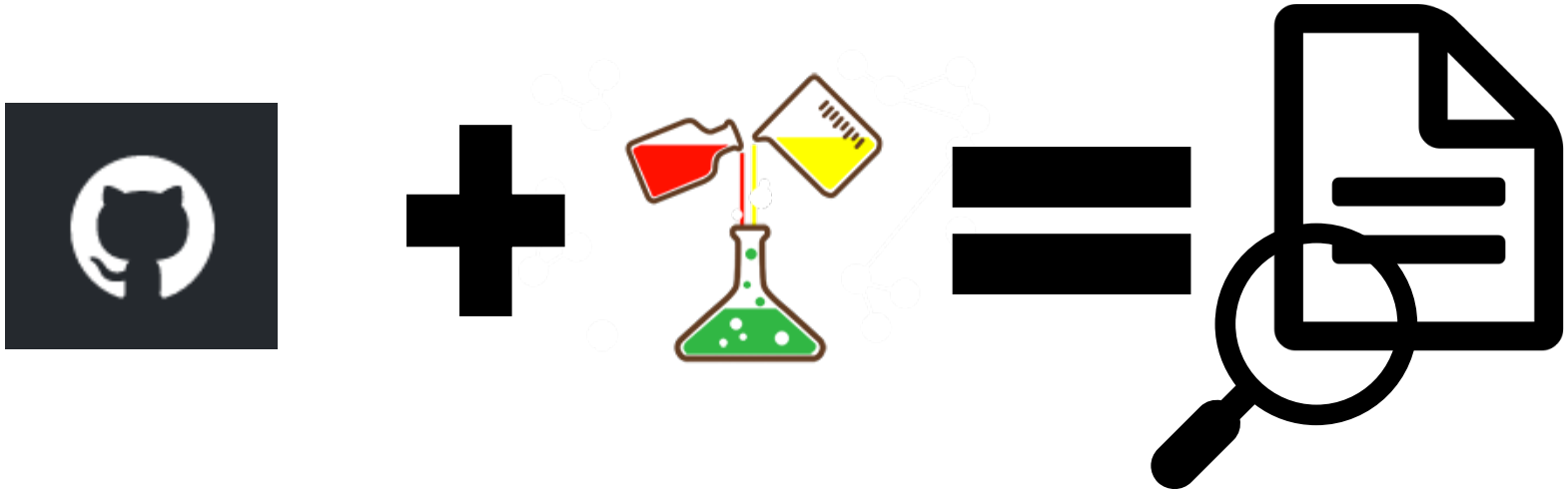


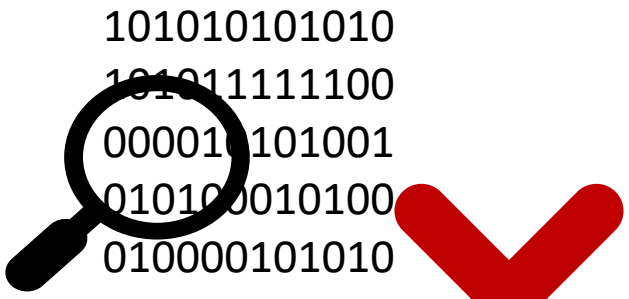
Understanding Code (Registry)

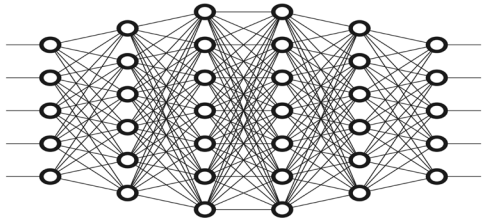
Extending Code (OPENED)



Orchestrating @Scale (Apiary)







Static Analysis

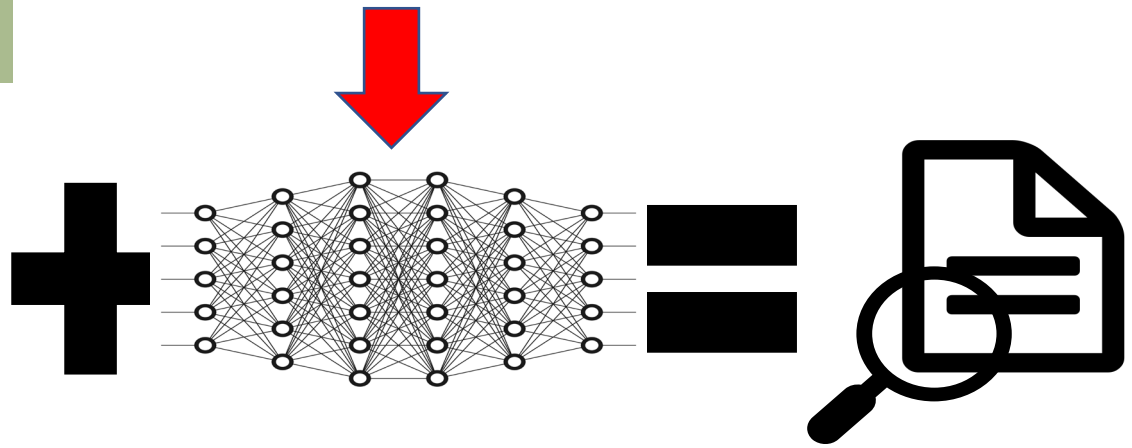
Machine Learning (LLMs)

- **Precise and Correct**
- **Hard to understand**
- **Fails to scale**

- **Potentially introduces errors**
- **Easy to understand**
- **Arbitrary scaling properties**

Input Granularity

Hyper Params



101010101010
101011111100
000010101001

101010101010
101011111100
000010101001

101010101010
101011111100
000010101001

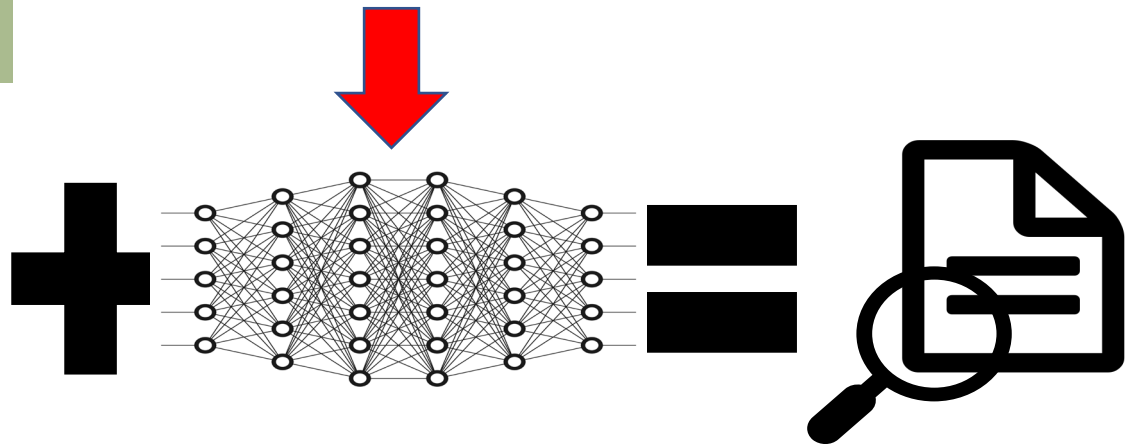
Input Granularity

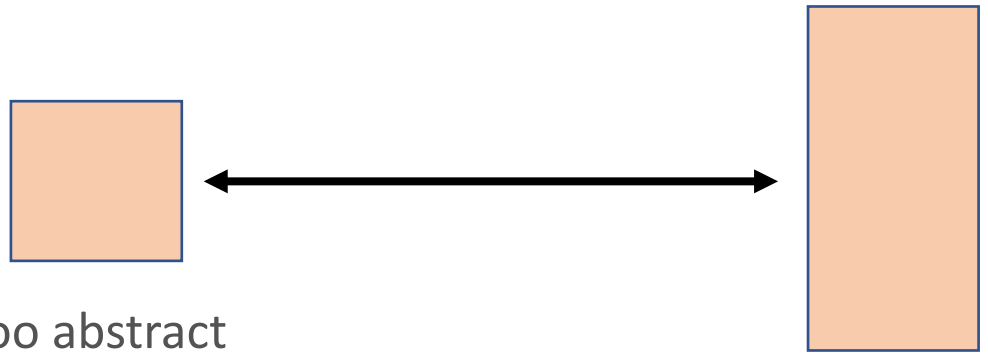
Hyper Params

101010101010
101011111100
000010101001

101010101010
101011111100
000010101001

101010101010
101011111100
000010101001



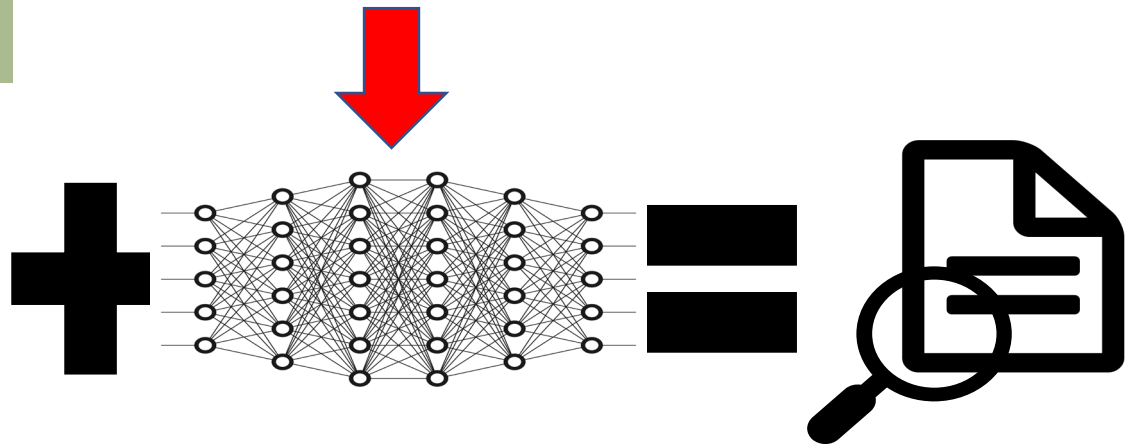


Too abstract

Too Verbose

Input Granularity

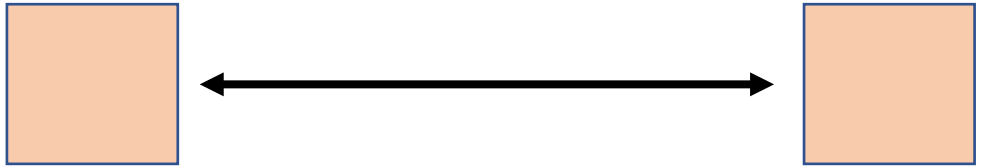
Hyper Params



101010101010
101011111100
000010101001

101010101010
101011111100
000010101001

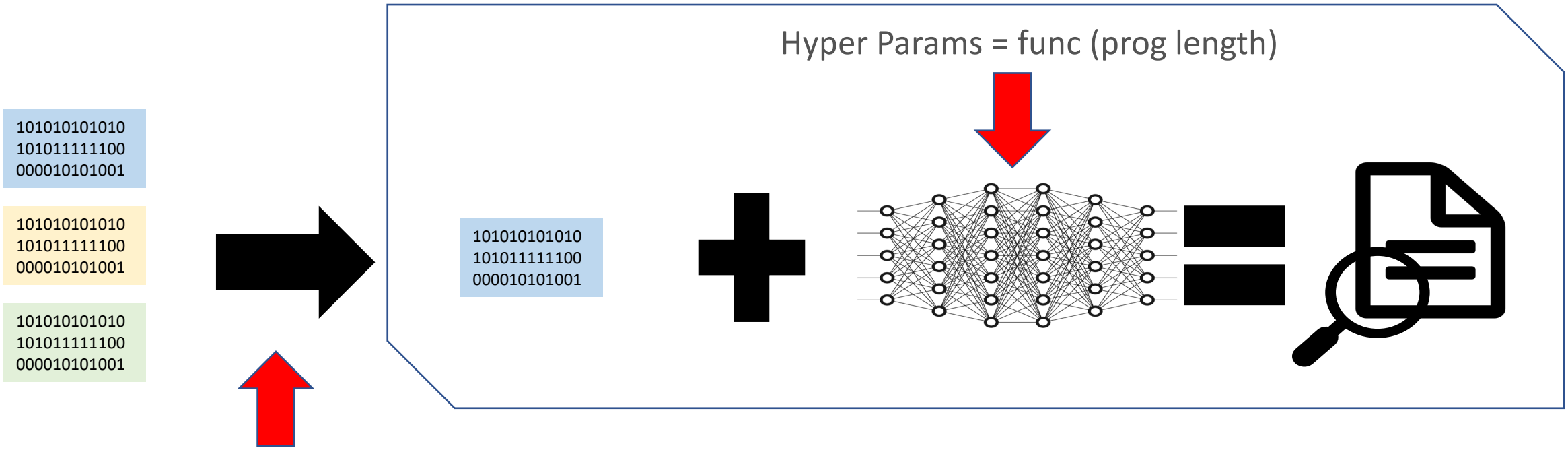
101010101010
101011111100
000010101001



Per file summary
(too convoluted)

Per line summary
(too mundane)

Explainability is on a Per Function Basis

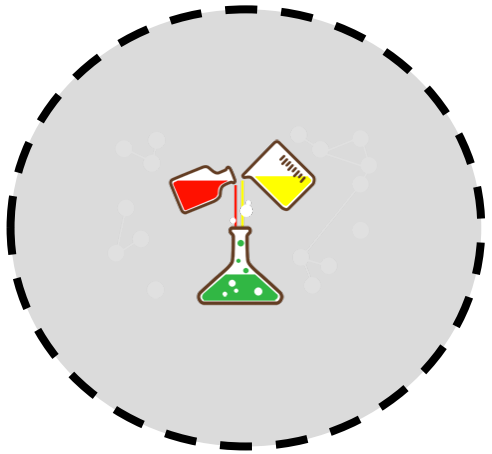


Decompose into functions

(eBPF)Registry: Simplify Code Introspection and Analysis

- Automated inference of code documentation
- Automated detection of code requirements
  - Enable duplicate code identification
  - Enable detection of conflicts
- Ongoing work: User-first documentation
  - Repository for all opensource eBPF programs
  - GUI + ElasticSearch for quick+advanced searches

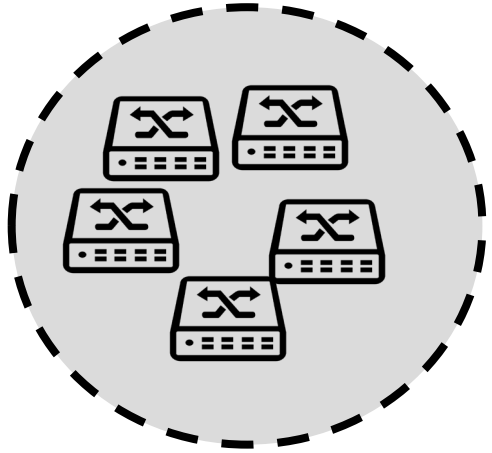
Roadmap



Background

Understanding Code (Registry)

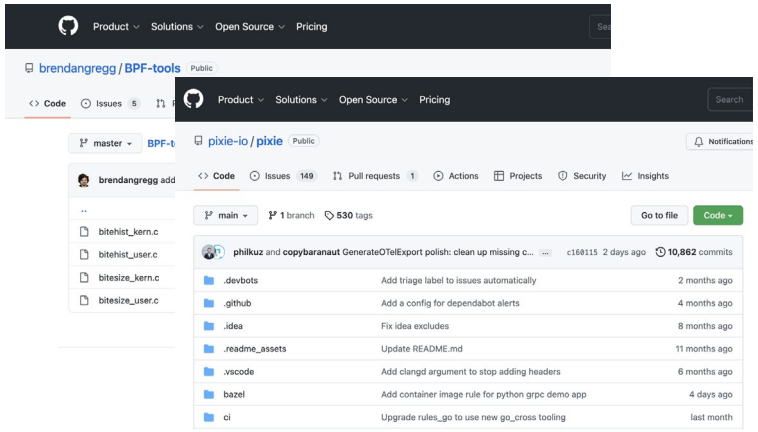
Extending Code (OPENED)



Orchestrating @Scale (Apiary)

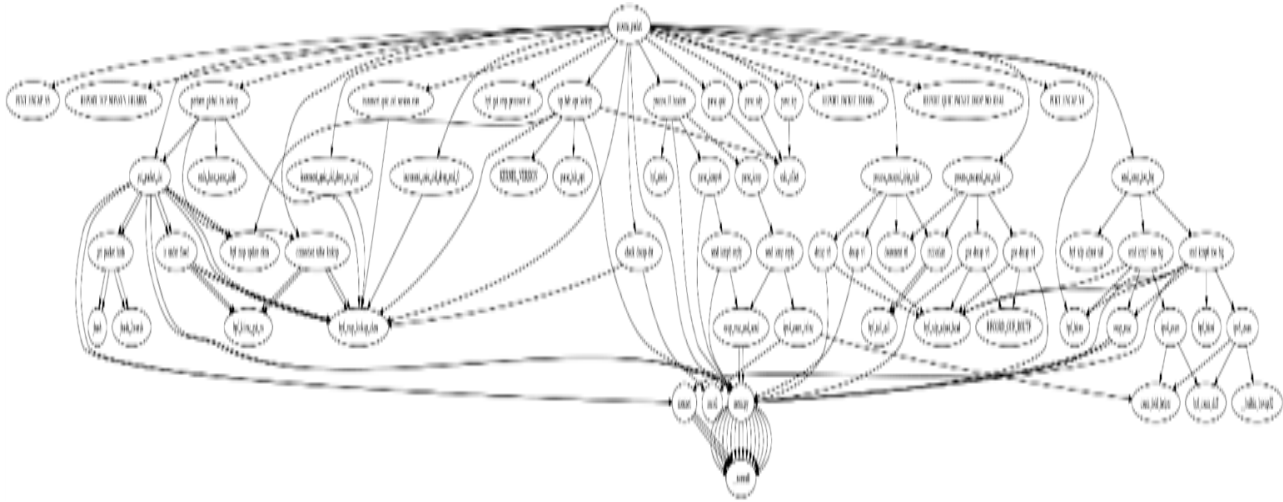
eBPF Programs are Monoliths

One Off Programs



Observability

Complex Codebase(s)



Network Functions

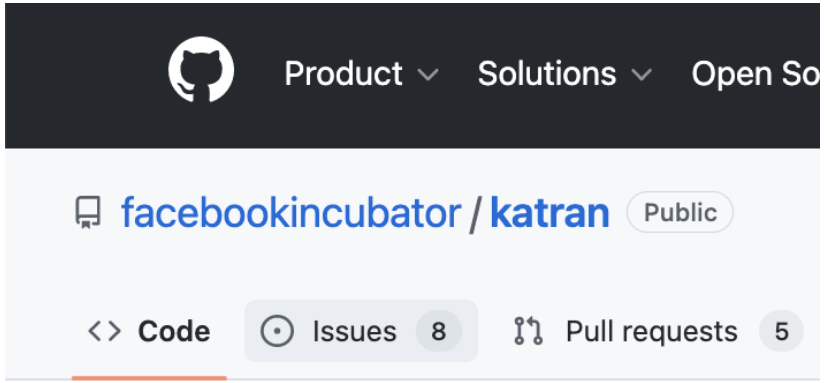
*Code from a Katran function

Implications of Monolith on Developer Productivity

Developing a new program

Find sub functionality on GitHub

Extracting and reusing functionality is non-trival

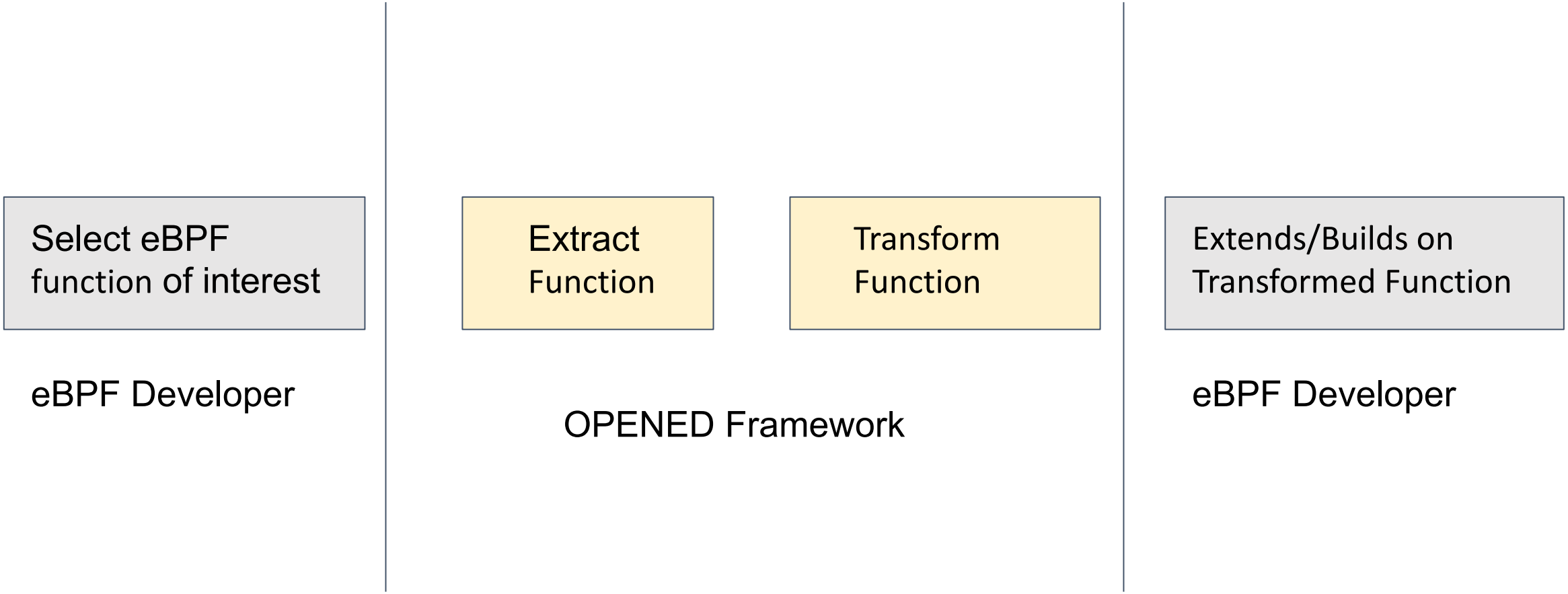


Step 1: Extract lines

Step 2: Identify + Extract Deps

Surprise Step 3: Rewrite for your target hookpoint

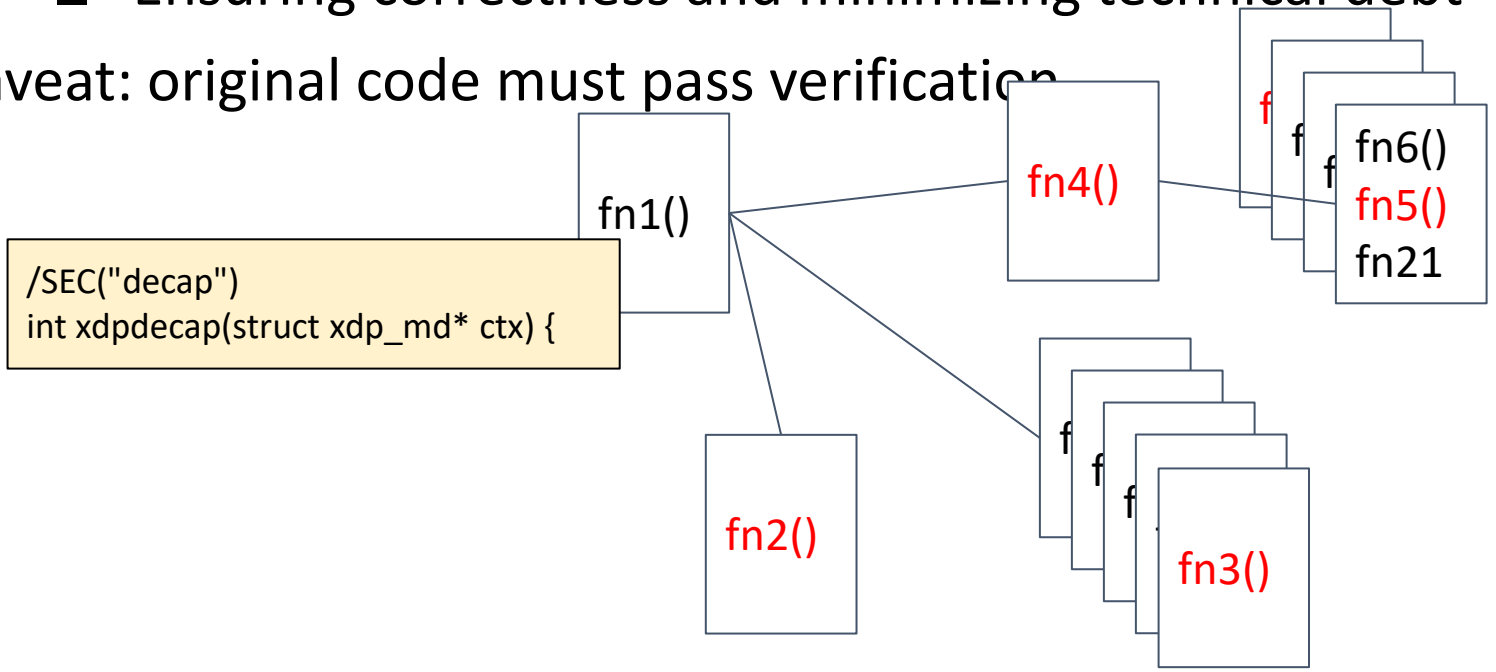
The OPENED Vision



Extraction

Extract eBPF func as an independently loadable module

- Identify all dependencies of the eBPF function
  - Dependencies: function call graph, Maps & associated structures, header files
  - Extract relevant dependencies while
    - Ensuring correctness and minimizing technical debt
- Caveat: original code must pass verification

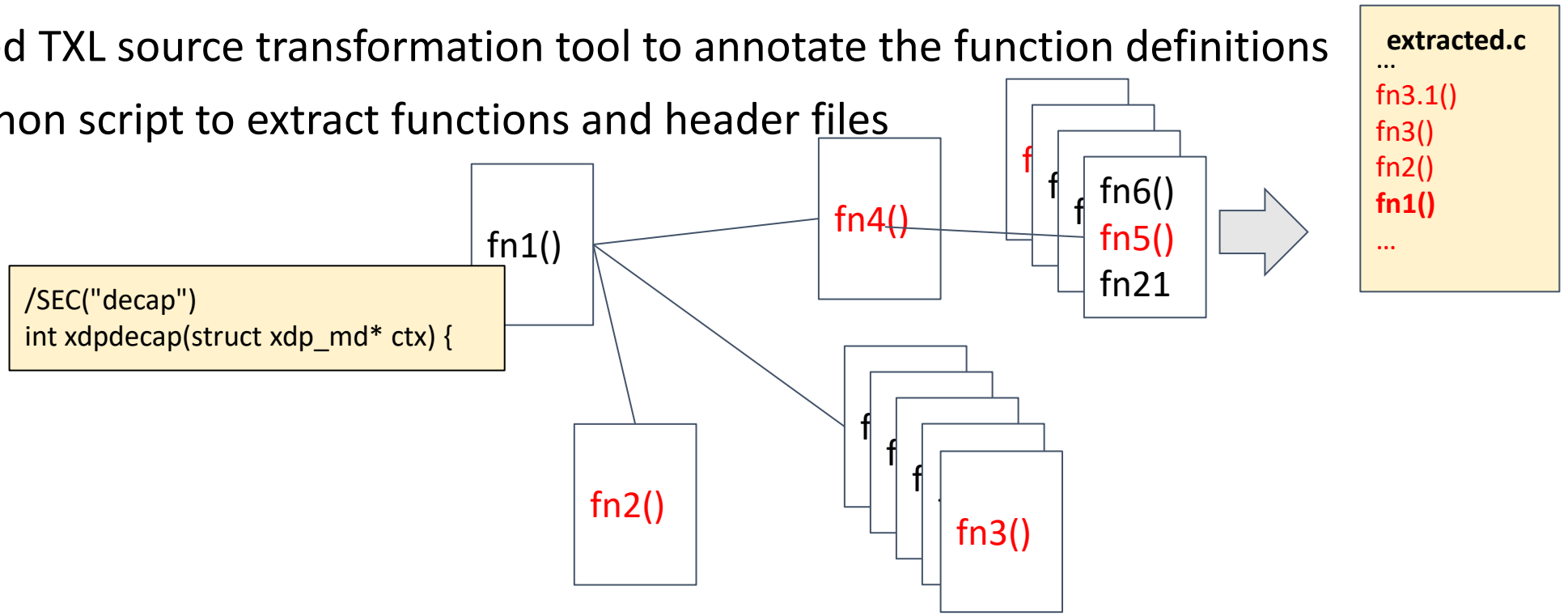


Challenge: Dependency Extraction

Function Dependencies

- Extended codequery tool[1],
  - Recursively identify function call graph
  - Uses cscope and ctags and sqlite internally
- Used TXL source transformation tool to annotate the function definitions
- Python script to extract functions and header files

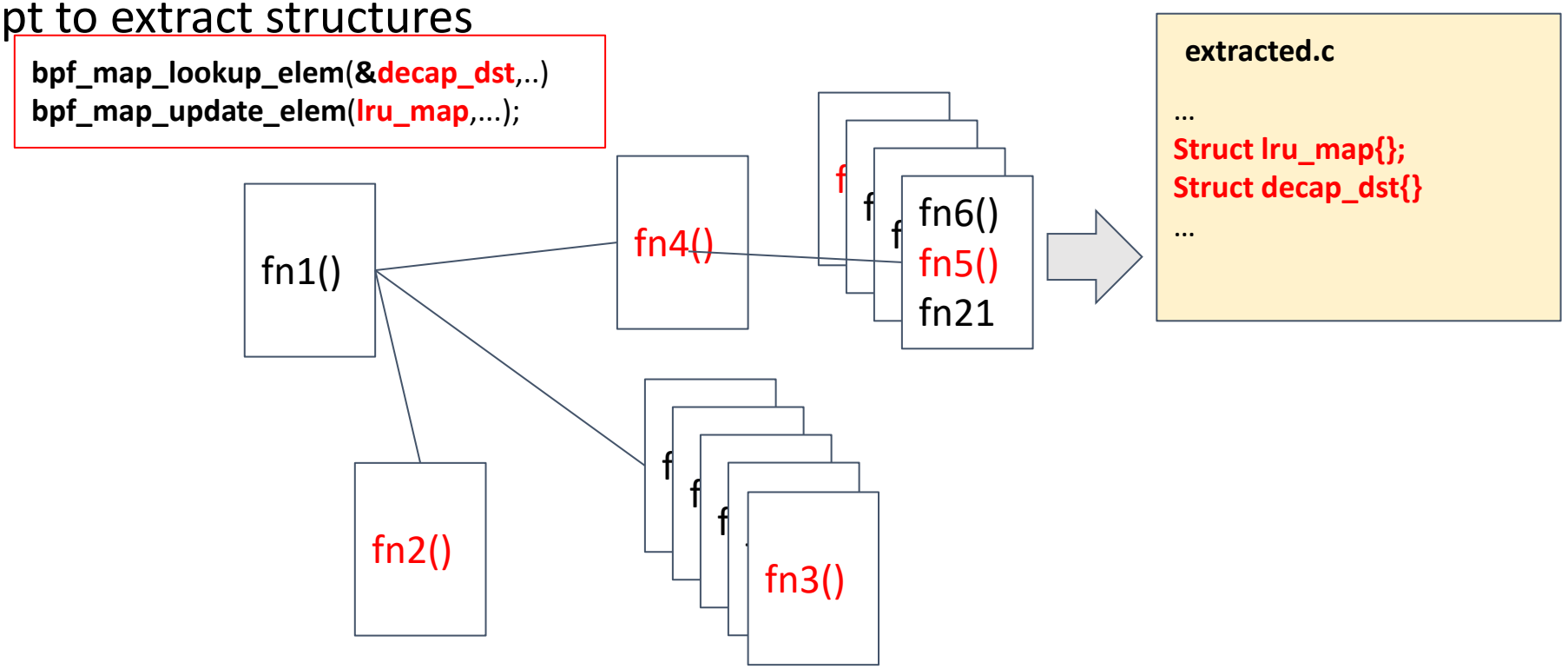
```
/* Extracted from
 /root/github/demo_lpc/ codequery/katran/decap_kern.c
startLine: 223 endLine: 247 */
SEC("decap")
int xdpdecap(struct xdp_md* ctx) {
```



Challenge: Dependency Extraction

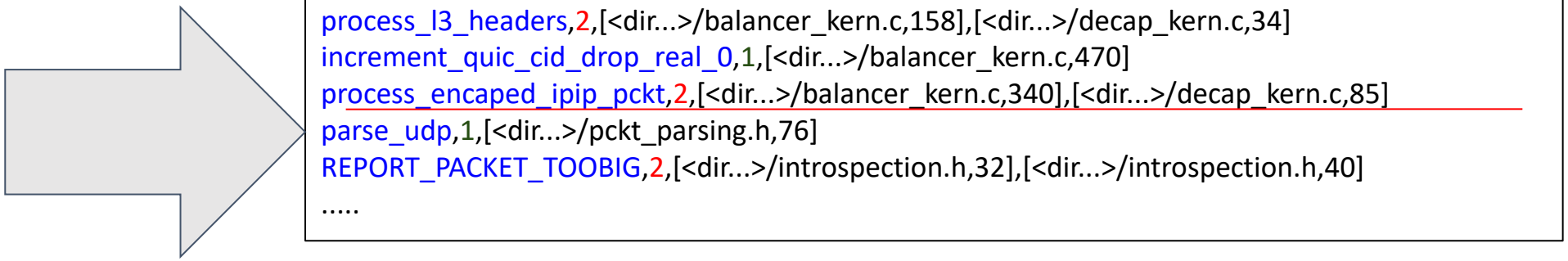
Map Definitions

- eBPF specific method of tracking bpf_map_update/lookup_elem while parsing call flow graph
- TXL code transformation tool to annotate maps and other data structures needed
- Python script to extract structures



Challenge: Dependency Extraction

Multiple declaration of dependencies (both maps & functions)



Annotated Function Call Graph

Preserve MACRO Definitions during Extraction

Identify and propagate preprocessor guards into extracted code

balancer_kern.c

```
#ifdef GLOBAL_LRU_LOOKUP

__attribute__((__always_inline__)) static inline bool
reals_have_same_addr(
   struct real_definition* a,
                  struct real_definition* b) {
  ...
  ...
}

__attribute__((__always_inline__)) static inline int
perform_global_lru_lookup(
   struct real_definition** dst,
   struct packet_description* pckt,..){
   ...
   ...
}

#endif // GLOBAL_LRU_LOOKUP
```

extracted.c

```
#ifdef  GLOBAL_LRU_LOOKUP
/* Extracted from balancer_kern.c  startLine: 261 endLine: 277 */
__attribute__((__always_inline__)) static inline bool
reals_have_same_addr(
   struct real_definition* a,
   struct real_definition* b) {
  ...
  ...
}
#endif
#ifdef  GLOBAL_LRU_LOOKUP
/* Extracted from balancer_kern.c  startLine: 279 endLine: 337 */
__attribute__((__always_inline__)) static inline int
perform_global_lru_lookup(
   struct real_definition** dst,
   struct packet_description* pckt,..){
   ...
}


#endif // GLOBAL_LRU_LOOKUP
```

Challenge: Minimize Code Debt

- Maintain ordering between definitions and invocations
- Propagate license into newly created c file (with extracted code).
- Identify and copy relevant current directory includes into extraction site.
  - Introduce preprocessor guards in new header files
- Rewrite Makefiles (currently Manual).

#include balancer_const.h”

#IFDEF BALANCER_CONST_ OPF

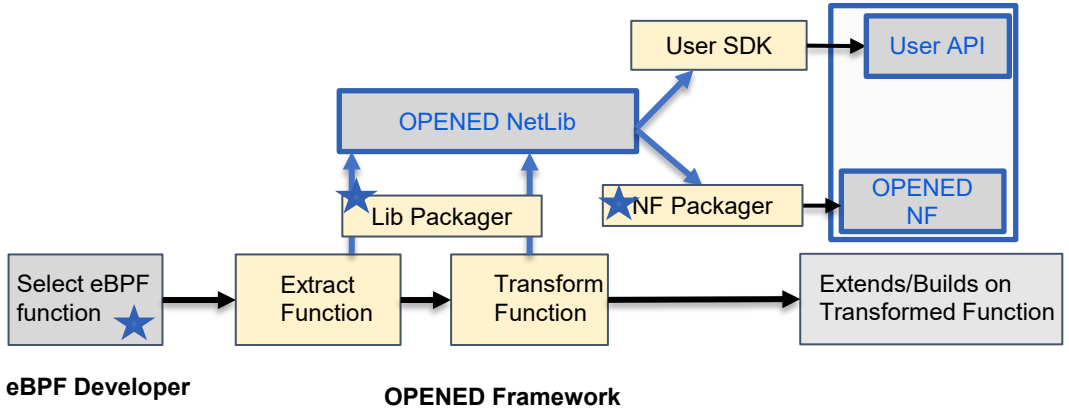
#include balancer_const.h”

#ENDIF

OPENED Vision: Reduce time to new functionality development

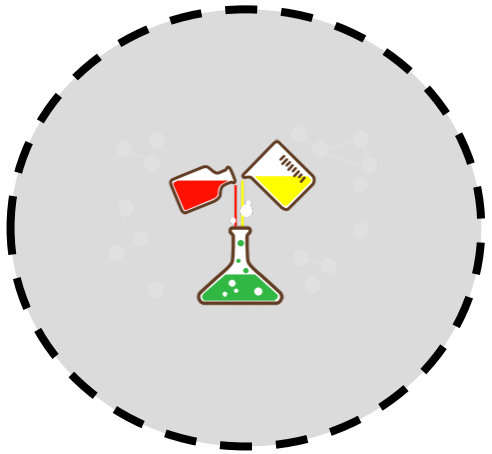
- Automated extraction of relevant code
- Automated transformation of code

- Ongoing work:
  - Create a library of common functions
  - Create interface for introspecting into deployed functions



Demo: https://lpc.events/event/16/contributions/1370/

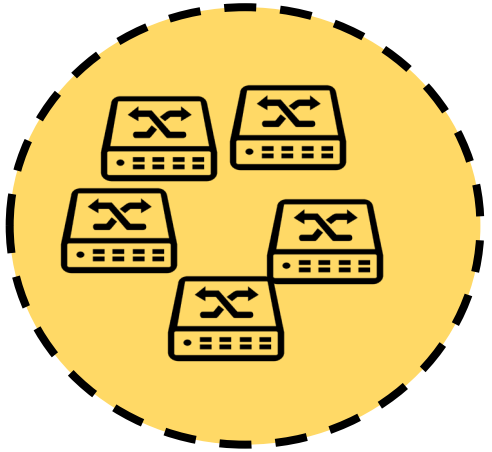
Roadmap



Background

Understanding Code (Registry)

Extending Code (OPENED)

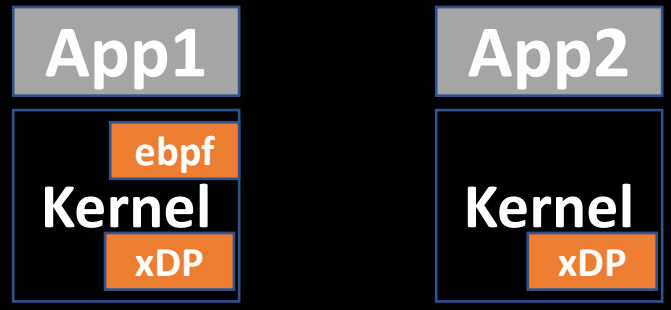


Orchestrating @Scale (Apiary)

Kubernetes is to Containers

As

?????? is to eBPFPr



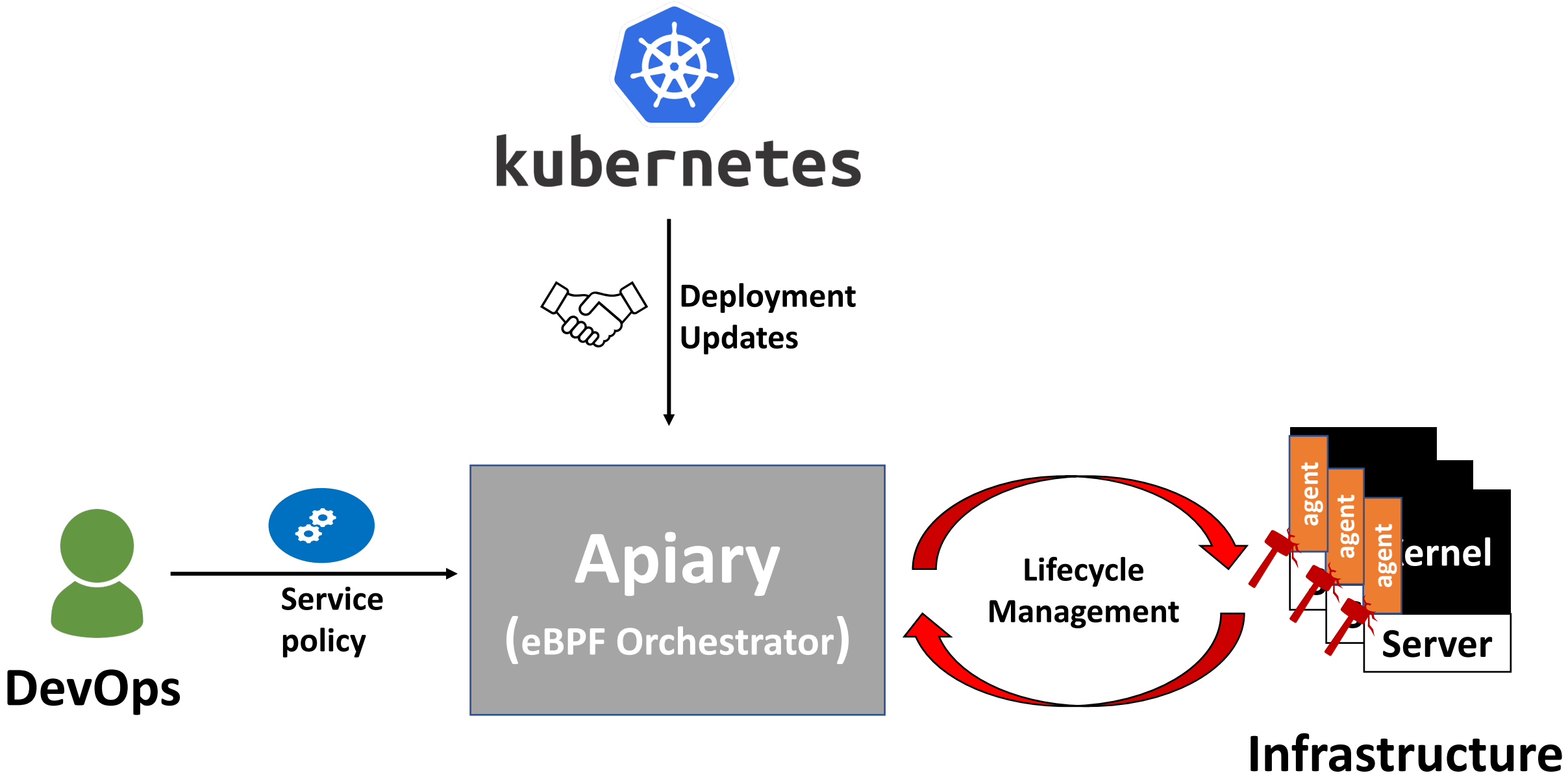
Unique program configurations

Constantly Deploy/undeploy

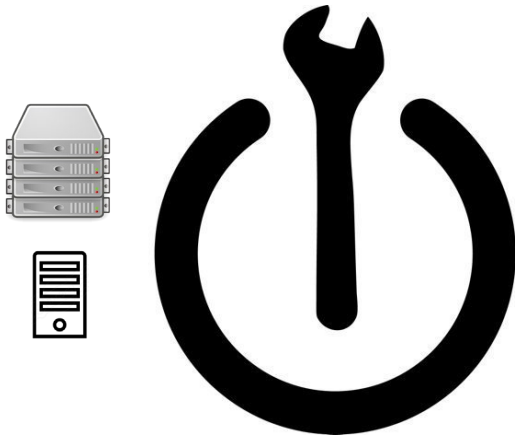
ployed programs = func (Apps, Role)

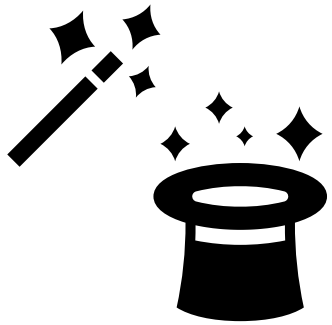
Infra Dynamics

App-Specific Policies


kubernetes
Deployment
Updates
DevOps
Service
policy
Apiary
(eBPF Orchestrator)
Lifecycle
Management
agent
agent
agent
ernel
Server
Infrastructure

Open Problems and Deployment Challenges





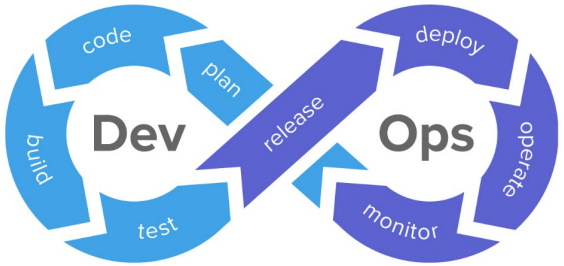
Code Portability

High level Language Abstractions

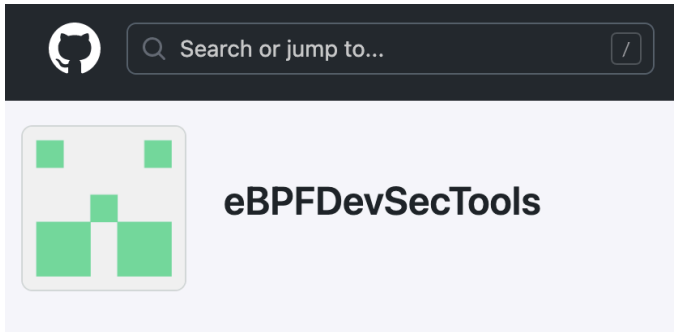
Debugging & Diagnosis

DevTools for Supporting @Scale Adoption of eBPF

eBPF is terraforming modern infrastructures



Ecosystem lacks support for largescale DevOps



Ensemble of tools for lifecycle management

Carnegie Mellon University

BROWN

IBM Research

Join Us!

QUEEN MARY UNIVERSITY OF LONDON

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

BU

भारतीय प्रौद्योगिकी संस्थान पटना
INDIAN INSTITUTE OF TECHNOLOGY PATNA

Submit your use cases for programs to be decomposed

https://github.com/eBPFDevSecTools