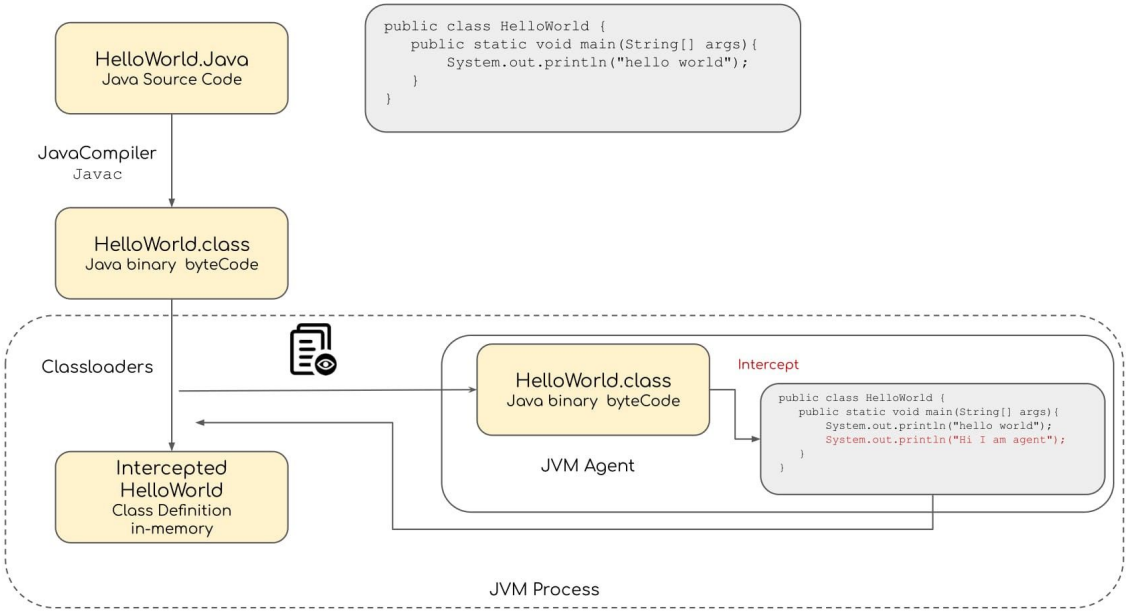
Instrument Go application using eBPF

Java agent

- Special class using Java instrumental API
- Instrumental Java application
- Inject code into bytecode

Java agent

-


HelloWorld.Java
Java Source Code
public class HelloWorld {
public static void main(String[] args){
System.out.println("hello world");
}
}
JavaCompiler
Javac
HelloWorld.class
Java binary byteCode
Classloaders
HelloWorld.class
Java binary byteCode
Intercept
public class HelloWorld {
public static void main(String[] args){
System.out.println("hello world");
System.out.println("Hi I am agent");
}
}
JVM Agent
Intercepted
HelloWorld
Class Definition
in-memory
JVM Process

Java agent

- https://docs.datadoghq.com/tracing/setup_overview/setup/java/?tab=containers
- https://github.com/open-telemetry/opentelemetry-java-instrumentation

Problems

Capture state of program, function arguments ?

- Solution: ???

Monkey patching

- Dynamically update the behavior of a piece of code at run-time.

Available in many dynamic type lang: javascript, python,... and some language with supported VM like: JVM, PHP

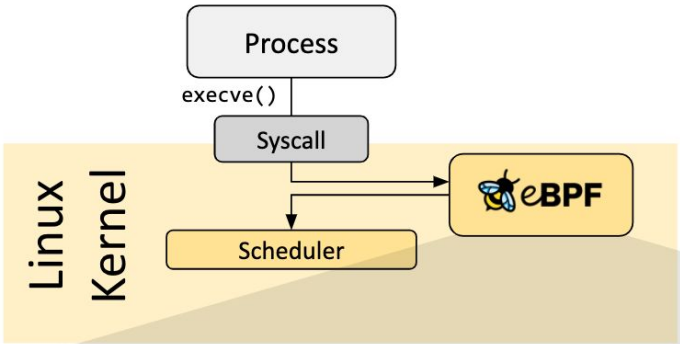
Go agent ? No ?

- There no JVM in Go
- Go compiled into binary, asm, not something bytecode like Java (almost same structure)

What

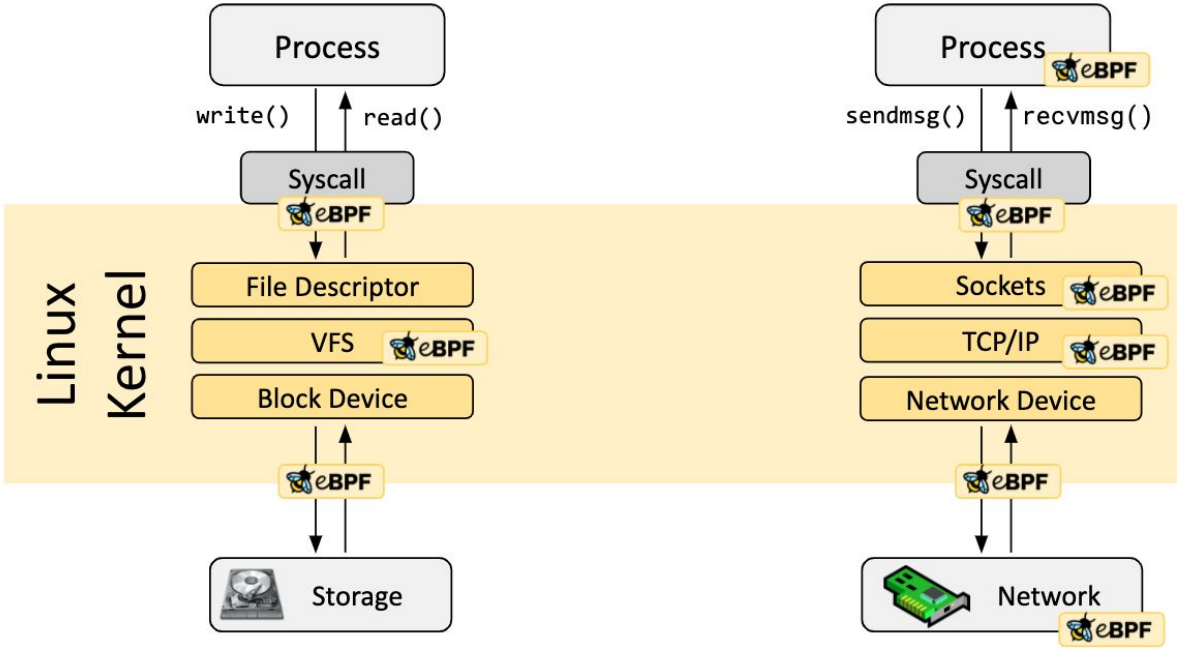
- is a feature of the Linux kernel (since Linux 4.x+).
- It’s a virtual machine inside the kernel
- allows the kernel to run BPF bytecode

What

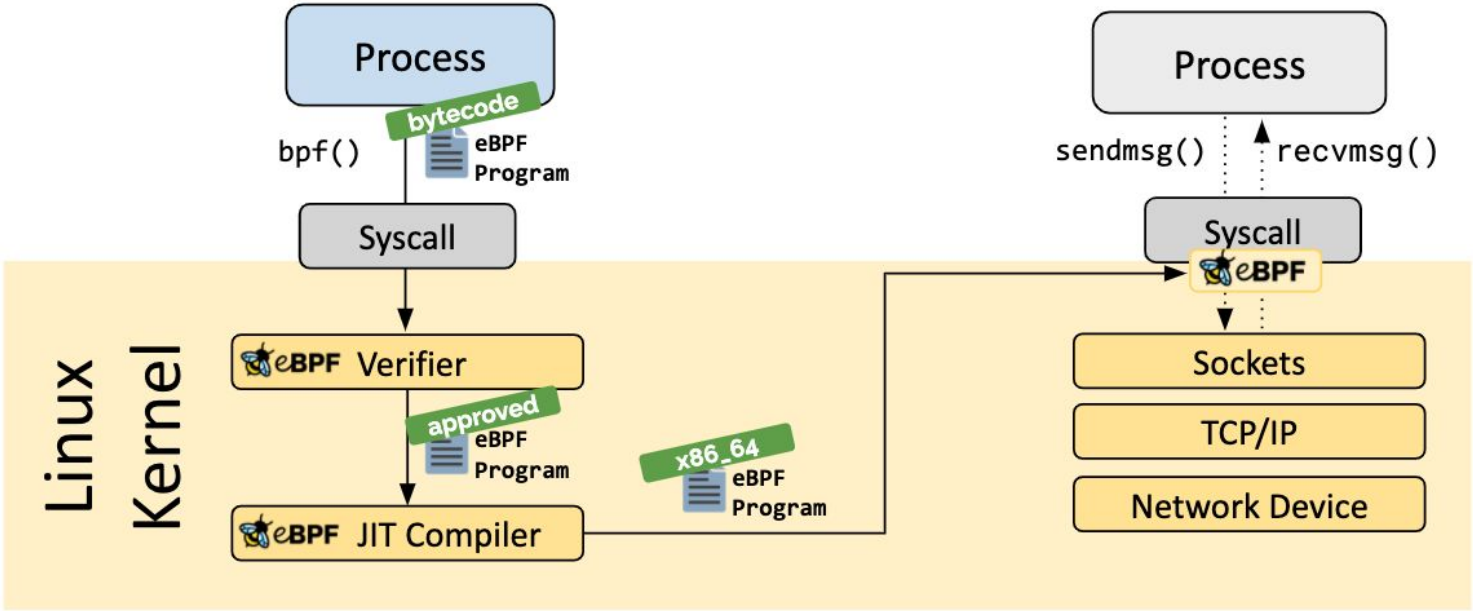



```c
int syscall__ret_execve(struct pt_regs *ctx)
{
        struct comm_event event = {
                .pid = bpf_get_current_pid_tgid() >> 32,
                .type = TYPE_RETURN,
        };

        bpf_get_current_comm(&event.comm, sizeof(event.comm));
        comm_events.perf_submit(ctx, &event, sizeof(event));

        return 0;
}
```

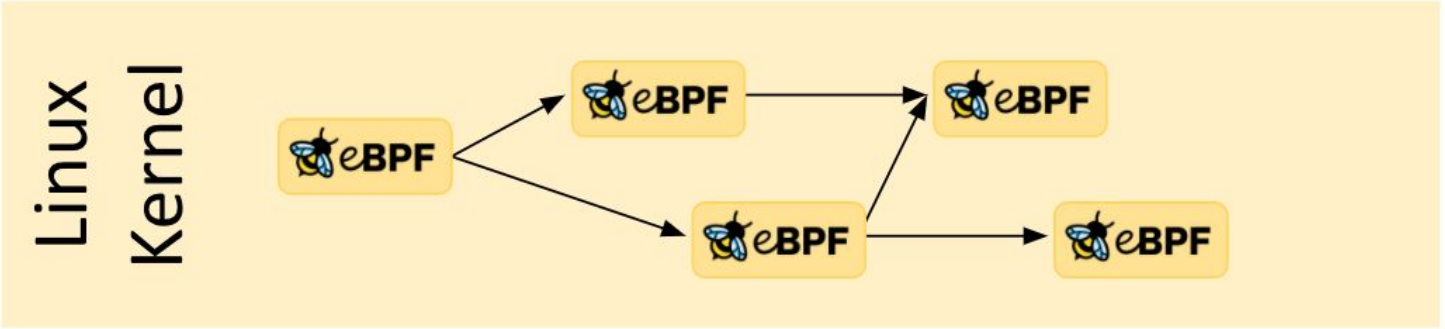
What


Process
write()
read()
Syscall
eBPF
File Descriptor
VFS
eBPF
Block Device
eBPF
Storage
Linux Kernel
Process
eBPF
sendmsg()
recvmsg()
Syscall
eBPF
Sockets
eBPF
TCP/IP
eBPF
Network Device
eBPF
Network
eBPF

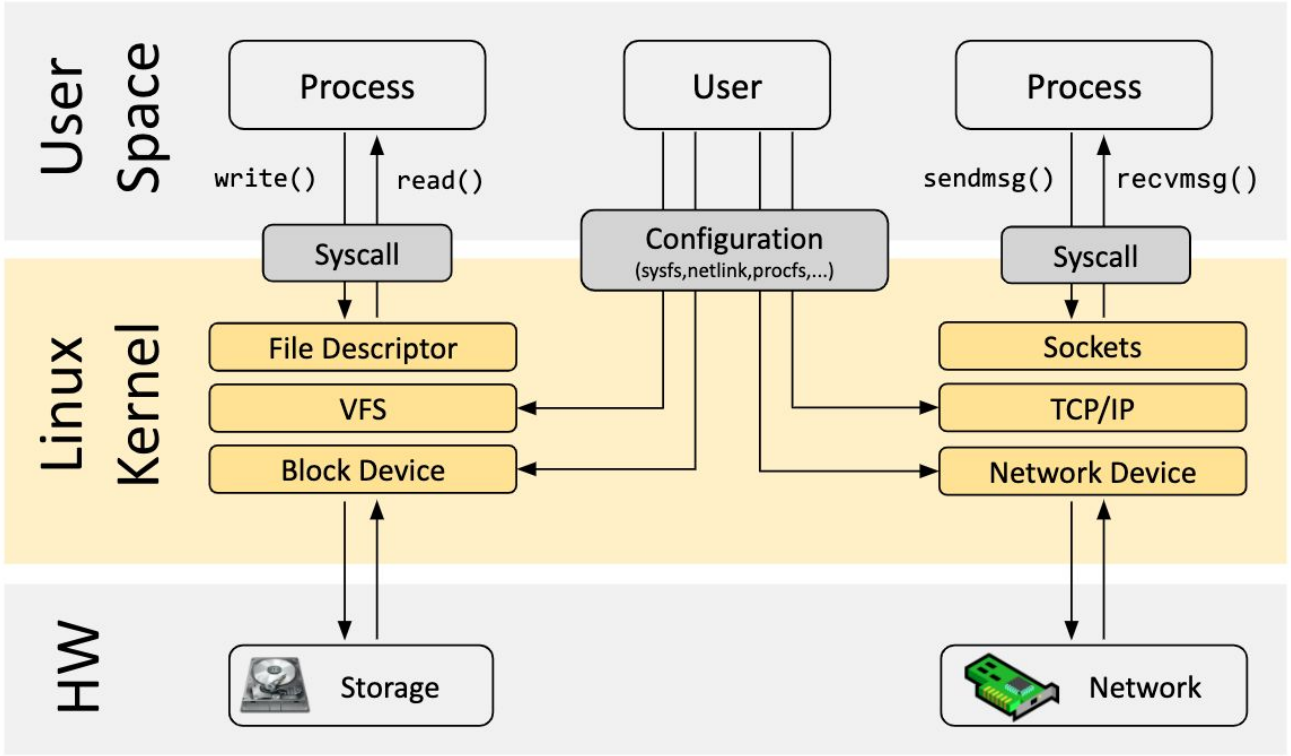
What


Process
bytecode
bpf()
eBPF
Program
Syscall
Linux
Kernel
eBPF Verifier
approved
eBPF
Program
eBPF JIT Compiler
x86_64
eBPF
Program
Process
sendmsg()
recvmsg()
Syscall
eBPF
Sockets
TCP/IP
Network Device

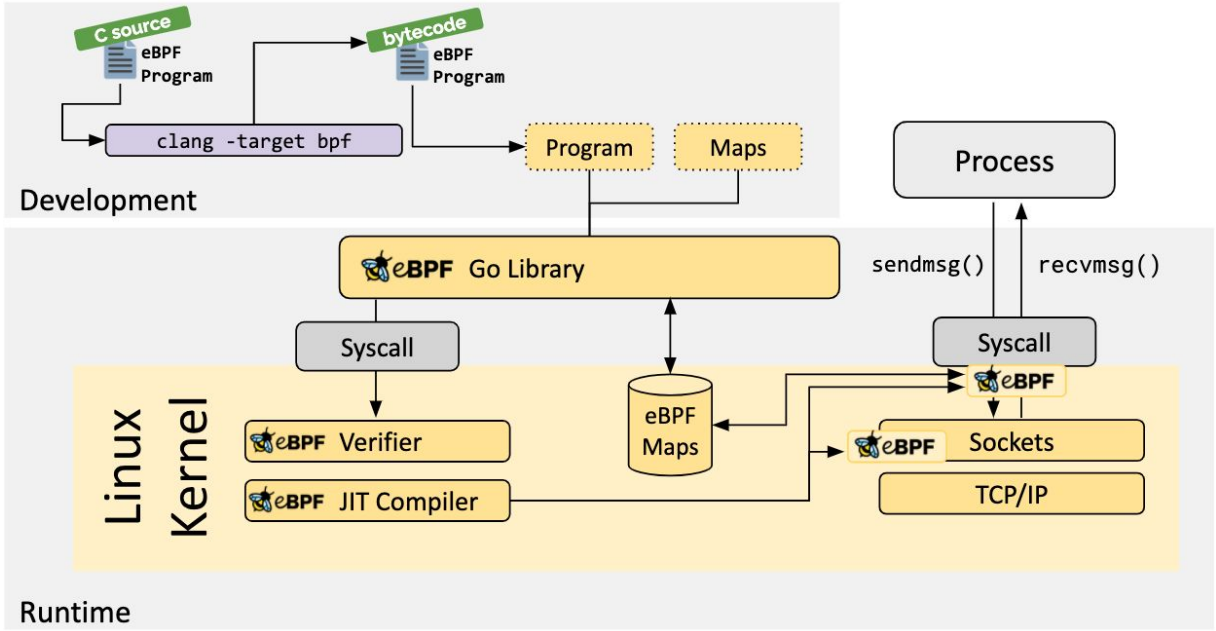
What



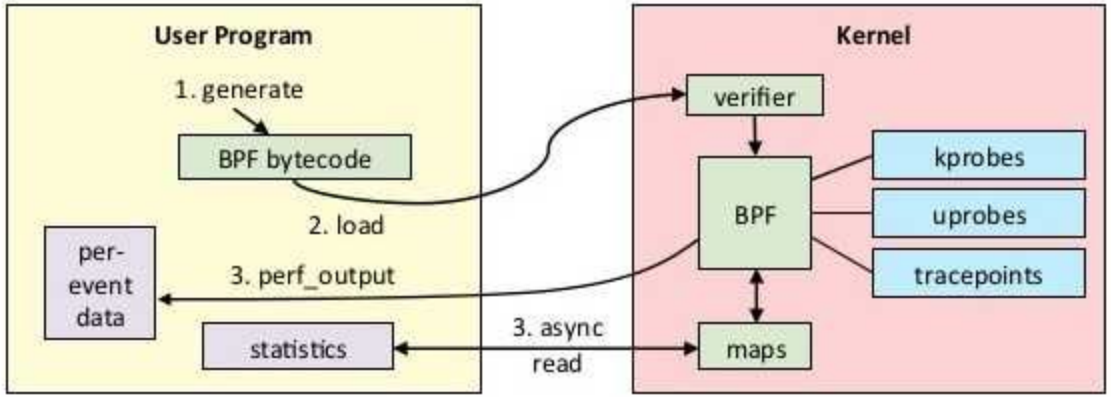
What


User Space
Process
User
Process
write()
read()
sendmsg()
recvmsg()
Syscall
Configuration
(sysfs,netlink,procfs,...)
Syscall
Linux Kernel
File Descriptor
VFS
Block Device
Sockets
TCP/IP
Network Device
HW
Storage
Network

What

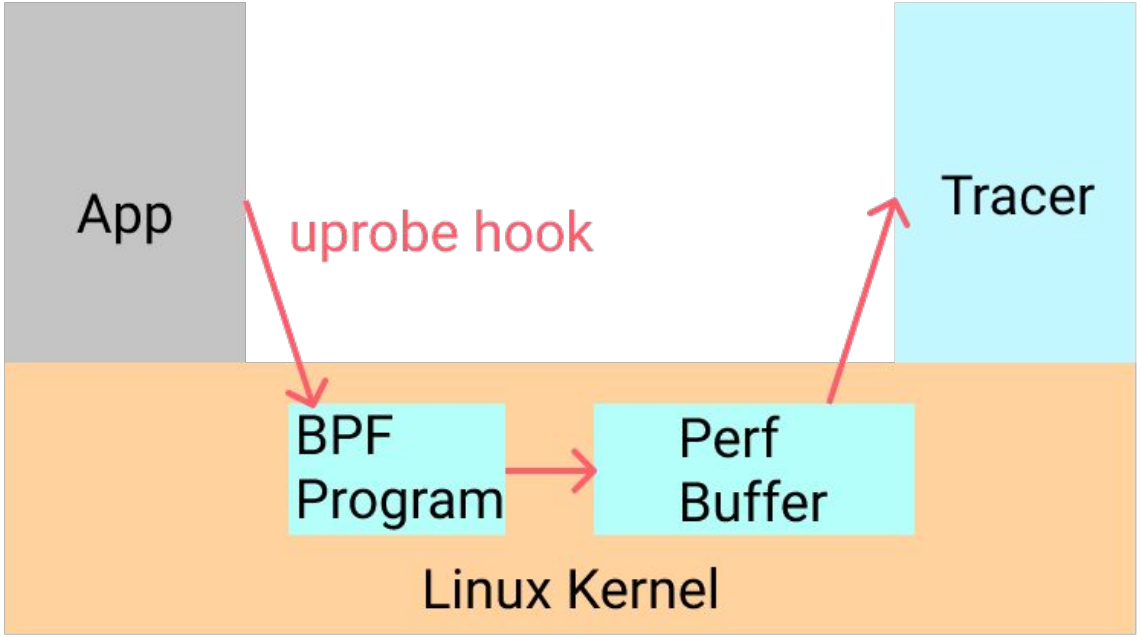

C source
eBPF Program
bytecode
eBPF Program
clang -target bpf
Program
Maps
Development
Process
eBPF Go Library
sendmsg()
recvmsg()
Syscall
Syscall
eBPF
eBPF Maps
eBPF Verifier
eBPF JIT Compiler
eBPF Sockets
TCP/IP
Linux Kernel
Runtime

What


User Program
1. generate
BPF bytecode
2. load
per-
event
data
3. perf_output
statistics
3. async
read
Kernel
verifier
BPF
maps
kprobes
uprobes
tracepoints

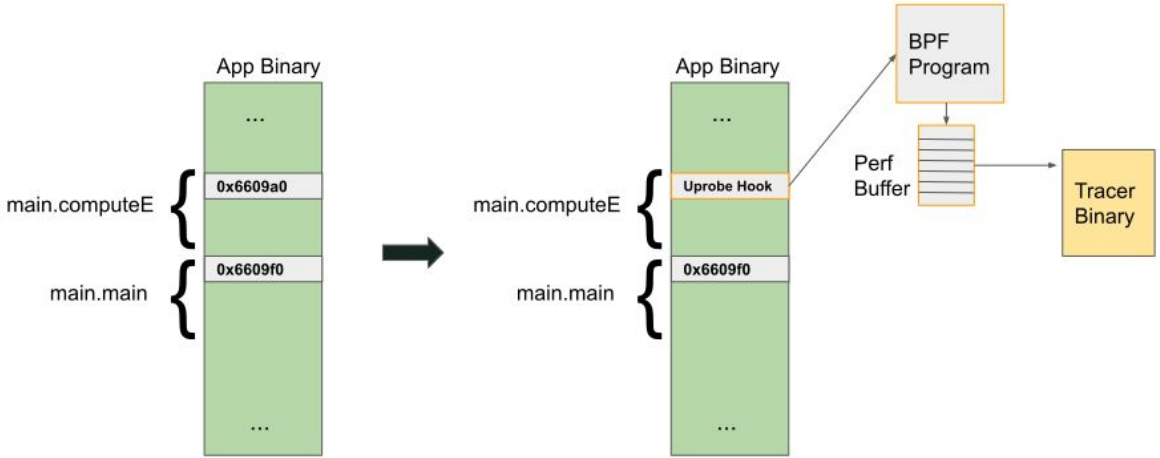
Demo

-


App
uprobe hook
Tracer
BPF
Program
Perf
Buffer
Linux Kernel

Demo

-


App Binary
...
main.computeE
0x6609a0
main.main
0x6609f0
...
App Binary
...
main.computeE
Uprobe Hook
main.main
0x6609f0
...
BPF
Program
Perf
Buffer
Tracer
Binary

Use-case

- Dynamic instrumental
- Tracing
- Dynamic logging
- …

References

https://docs.oracle.com/javase/1.5.0/docs/api/java/lang/instrument/package-summary.html

https://www.cncf.io/blog/2021/11/17/debugging-with-ebpf-part-1-tracing-go-function-arguments-in-prod/

https://nakryiko.com/posts/bpf-ringbuf/#:~:text=Perfbuf%20is%20a%20collection%20of,memory%20and%20event%20re%2Dordering.