Fast Packet Processing with eBPF and XDP:
Concepts, Code, Challenges and Applications
요약 정리

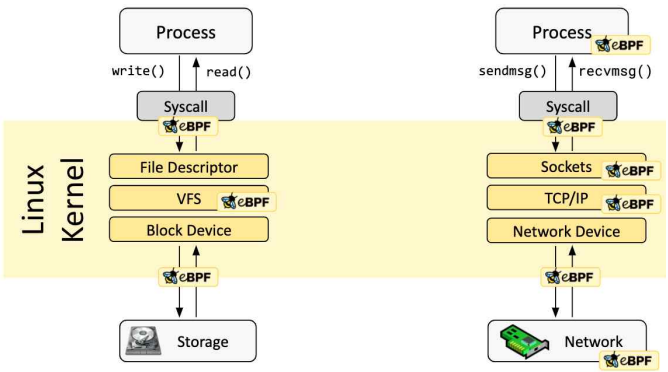
[18반 라민우]

(!) 격주 보고서 작성에 필요한 내용만을 포함하기 위해 eBPF/XDP에 대한 설명을 최대한 간략하게 정리하고 핵심 정보를 중심으로 서술했다.

▶ eBPF란?
다양한 명령어 체계 , 실행 환경(VM 환경)을 리눅스 커널에서 제공하는 시스템
커널 단에 프로그래머가 원하는 프로그램을 적용할 수 있게 하자는 것이 기본적인 목적이다.
BPF는 cBPF(구버전), eBPF 로 나뉠 수 있으며, eBPF 기준 레지스터, 스택, 맵을 구현한 BPF machine 체계를 갖추고 있다.

▶ eBPF 프로그램의 실행 과정
[user space] 에서의 기본적인 과정을 설명하자면, 고수준 프로그래밍 언어를 통해 코드를 구성하고, LLVM 을 통해 프로그램을 커널에서 작동할 코드를 생성한다. [user space] 에서의 동작에서 있어서 핵심은 컴파일이라고 보면 된다.
적용할 코드를 실제로 커널 단에 삽입하고 실행하는 과정에서는 ELF eBPF loader 가 관여하는데, 컴파일을 통해 생성된 eBPF ELF 파일에 내재된 일련의 메타 데이터가 제공해주는 정보를 토대로 동작한다.
[kernel space] 에서는 위 [user space] 로부터 받은 BPF 코드를 실제로 동작을 요구할 커널 요소에 적용하기 위한 일련의 검증 & 컴파일 과정을 실시한다. 커널 공간에 삽입할 코드에 대한 검증은 'Verifier'가 담당하며 관련 알고리즘은 엄격한 기준을 갖고 있다는 점이 특징이다. 이후 'JIT compiler'를 통해 kernel 단에서 작동할 수 있는 기계어로 변환하면, 실제로 eBPF 프로그램을 원하는 커널 요소(우리는 XDP 구현을 위해 Network Device Driver에 연결한다) 에 본격적으로 부착할 수 있게 된다.

▶ 아래의 그림을 통해 eBPF의 다양한 활용 경로를 시각적으로 확인할 수 있음.
(단, eBPF/XDP의 실질적인 동작 과정에 대한 설명보다는 eBPF의 다양한 활용 가능성을 보여주는 간단한 지표로써 보는 것이 맞아보임)



▶ XDP 란?
eXpress Data Path, eBPF 안에 포함된 프로그램 타입중의 하나로, 전통적인 Packet Data를 처리하는

것에서 벗어나 특별한 방법으로 접근하는 방식이라고 보면 된다.
Packet Data를 처리함에 있어서 운영체제의 방식은 다양한 기능을 제공하지만, 그만큼 처리 과정의 효율성과 같은 부분을 재고할 여지가 있다. 이에 따라 kernel bypass를 구현한 기능(정확히는 toolkit)이 구현된 바도 있지만, 다양한 모듈을 제공받는 방식에서 벗어나 다이나믹하게 특정 하드웨어 장치와 연결하는 방식은 그만큼 포기해야하는 기능도 있다는 점이다.
이러한 단점을 극복하면서 효율적인 Data 처리 방식을 고안한 것이 XDP 라고 보면 된다.

▷ 논문의 방향성은 ?
기본적인 eBPF / XDP 에 대한 설명과 함께 간단한 코드 실행과 관련 프로젝트들을 나열하는 형태이다.
가이드라인을 제시하는 형태기 때문에, 실제로 eBPF/XDP에 대해 자세히 알고 싶다면,
위에 설명된 XDP, LLVM, kernel bypass 와 같은 키워드들은 다른 논문을 참고하는 것을 추천한다.

eBPF KEYWORD
eBPF, XDP, Virtual Machine
Verifier, JIT
LLVM