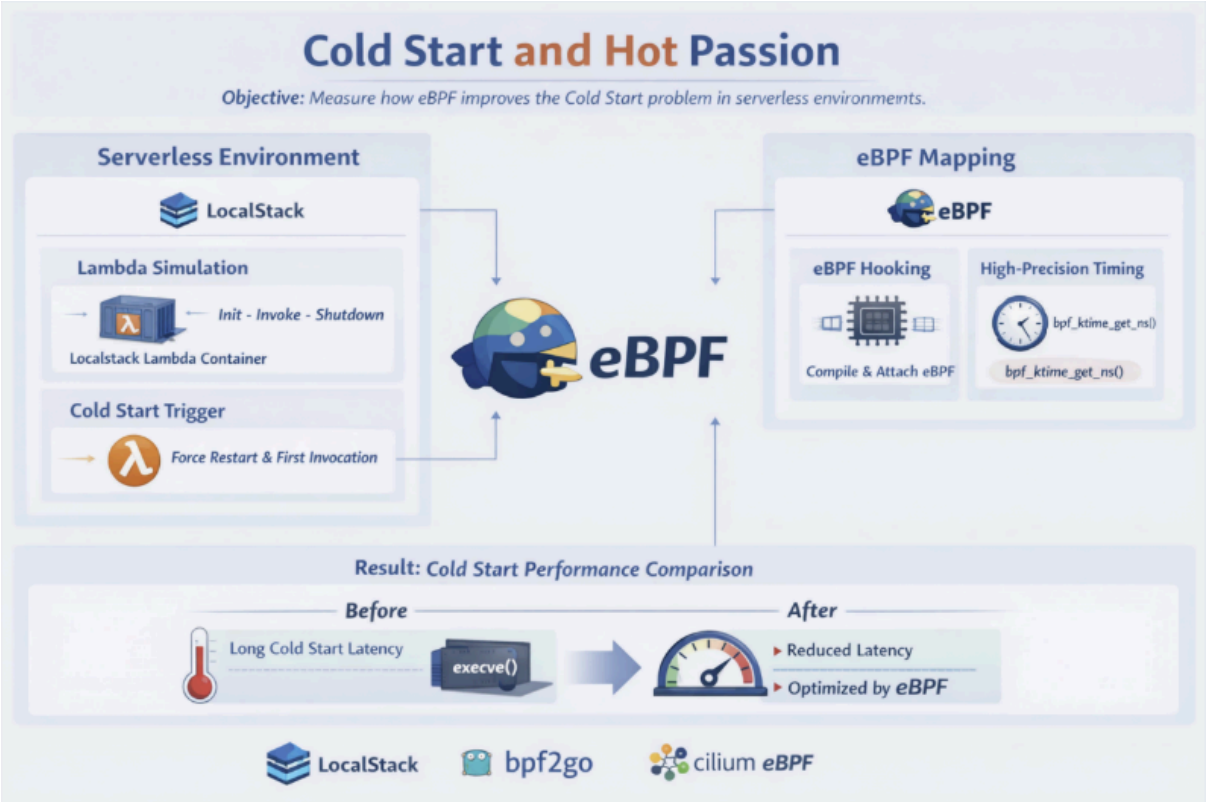
F4: Cold_Start_Hot_Passion
팀원: 김선희 김준엽 오진우 이예은

Zero-Copy Serverless Optimization using eBPF & Shared Memory

이 프로젝트는 eBPF와 호스트 공유 메모리 (`/dev/shm`)를 활용하여 서버리스 환경 (AWS Lambda)에서 반복적으로 발생하는 Cold Start 문제와 데이터 I/O 지연을 완화하기 위한 방안을 제안하고, 그 성능을 실험적으로 검증하는 것을 목적으로 한다. 특히 기존 네트워크 I/O 기반 방식과 Zero-copy Shared Memory 기반 방식을 비교함으로써, 지연 시간 감소 효과와 처리 효율 향상 가능성을 확인하고자 하였다.



Project Structure

```
.
├── baseline_src/          # [대조군] 기존 네트워크 I/O 기반 Lambda 환경
│   ├── handler.py         # 경량 S3 핸들러 (urllib SigV4 구현)
│   ├── Dockerfile
│   └── docker-compose.yml  # MiniStack 인프라 설정
├── shm_src/               # [실험군] Zero-copy Shared Memory 기반 환경
│   ├── handler.py         # zero-copy 핸들러 (pyarrow/mmap 기반)
│   ├── ringbuffer.py      # eBPF 통신용 링버퍼 모듈
│   ├── shm_init.py        # 공유 메모리 초기화 스크립트
│   ├── entrypoint.sh
│   ├── Dockerfile
│   └── docker-compose.yml  # Host IPC 및 /dev/shm 공유 설정 포함
├── ebpf_writer.py         # eBPF 기반 커널 이벤트 트레이서 및 SHM 기록기
├── headers/               # eBPF 컴파일용 C 헤더 파일들
├── scripts/               # 환경 세팅 자동화 쉘 스크립트
├── web_impl/              # Streamlit 기반 성능 비교 대시보드
└── dummy_data/            # 테스트용 S3 페이로드 데이터
```

How To Start

Ubuntu 22.04+, python 3.11 권장

1. 시스템 의존성 설치

eBPF 실행을 위한 커널 도구와 컨테이너 환경을 구축한다.
web_impl/prep_for_web.sh, call_base_func.sh, call_shm_func.sh, deploy_base_lambda.sh, deploy_shm_lambda.sh, setup_s3_data.sh 의 최상단 경로는 개인이 해당 레포지토리를 git clone 받은 경로로 변경해야 한다.

```
# 0. 가상환경 세팅
sudo apt-get update
sudo apt install python3-venv
python3 -m venv <가상환경 이름>
source <가상환경 이름>/bin/activate # 가상환경 진입

# 1. 커널 헤더 및 bpfcc-tools 설치
chmod +x scripts/*.sh
sudo ./scripts/set_ebpf.sh

# 2. Python 라이브러리 및 awslocal 설치
bash ./scripts/set_zerocopy.sh
pip install awscli-local
```

이 과정에서 externally-managed-environment 에러 발생 시 다음 명령어를 입력한다.

```
export PATH=$PATH:/home/ubuntu/.local/bin
python3 -m pip config set global.break-system-packages true
```

2. eBPF kernel shared memory 및 tracer 가동

시스템 콜을 감시하고, 감지된 이벤트를 공유 메모리에 기록하는 프로세스를 실행한다. 이것은 별도의 터미널 창에서 수행해야 한다.

```
# shared memory path 구축
python3 shm_src/bake_arrow_shm.py

# 커널 레벨 추적을 위해 sudo 권한 필요
sudo python3 ebpf_writer.py
```

3. MiniStack infra 구동

호스트의 IPC 네임스페이스와 /dev/shm을 공유하도록 설정된 실험군 환경을 실행한다.

```
cd shm_src
sudo docker-compose up -d
```

4. Lambda function 빌드 및 배포

각 환경의 Docker 이미지를 빌드하고 로컬 클라우드 환경에 등록한다.

```
# web_impl 디렉토리의 배포 스크립트 활용
cd ../web_impl

# 기존 자원 정리 및 함수 배포
bash clear_before_start.sh
bash prep_for_web.sh
```

5. 성능 비교 대시보드 실행

실시간으로 지연 시간을 비교하는 웹 인터페이스를 구동한다. 'ngrok http <포트>'는 다른 터미널 창에서 실행해야 한다.

```
# 대시보드 의존성 설치
bash ngrok.sh
sudo apt install nodejs -y
sudo apt install npm -y
npm install

# 실행
node server.js
ngrok http <포트>
```

How to Benchmark

1. https://zain-acosmistic-olen.ngrok-free.dev 접속
2. 좌측 패널의 "100회 자동 테스트" 클릭
3. 우측 패널에서 Baseline (Network I/O) vs SHM (Zero-copy)의 성능 차이 실시간 확인

References

- cilium/ebpf (https://github.com/cilium/ebpf)
- ministack (https://github.com/Nahuel990/ministack)