[EMDC OS]

Docker container 환경에서

이종가속기 추상화를 위한 공용 런타임

버전	내용
V1.0	최초 배포판
v0.1	초안 작성

성균관대학교 컴파일러 및 시스템 연구실

1. 개요

2. 배경 지식

    1) Intel DPC++ runtime plugin interface

    2) Software stack for accelerator computing

    3) Runtime and device driver for docker container

3. Native 환경에서 공용 런타임

    1) Intel CPU

    2) Intel FPGA PAC

    3) NVIDIA GPU

4. Docker container 환경에서 공용 런타임

    1) Runtimes

    2) User-space device driver

참고문헌

1. 개요

본 문서는 Intel DPC++를 활용하여 컨테이너 환경에서 이종가속기 추상화를 위한 공용 런타임에 대한 기술문서이다. Intel DPC++는 OpenCL runtime뿐만 아니라 device 특화 runtime도 확장할 수 있는 구조를 제공하며, Codeplay에서 NVIDIA CUDA runtime을 인터페이싱 할 수 있는 플러그인을 제공한다.

Intel CPU, Intel FPGA, NVIDIA GPU를 추상할 수 있는 공용 런타임을 native 환경에서 구축하고, docker container 환경에서도 구축하여 Apache OpenWhisk에서 테스트한다. 이때 docker container는 kernel-mode driver를 host와 공유하고 그 외 user-space의 소프트웨어는 별도로 설치해야 한다.

2. 배경 지식

1. Intel DPC++ runtime plugin interface

DPC++ runtime library의 SYCL API와 이종가속기의 native runtime 사이의 인터페이스 레이어이다. 기본적으로는 SYCL API가 OpenCL API를 호출하도록 추상화 DPC++ runtime과 OpenCL runtime을 연결한다. 그 외 다른 native runtime도 지원할 수 있도록 확장 가능한 구조를 제공한다. 이러한 구조를 이용하여 Codeplay에서 NVIDIA CUDA runtime을 사용할 수 있도록 개발하였고, Intel DPC++에 병합되어 있다. 그림 1은 Intel DPC++ runtime의 전체 stack을 나타낸다.

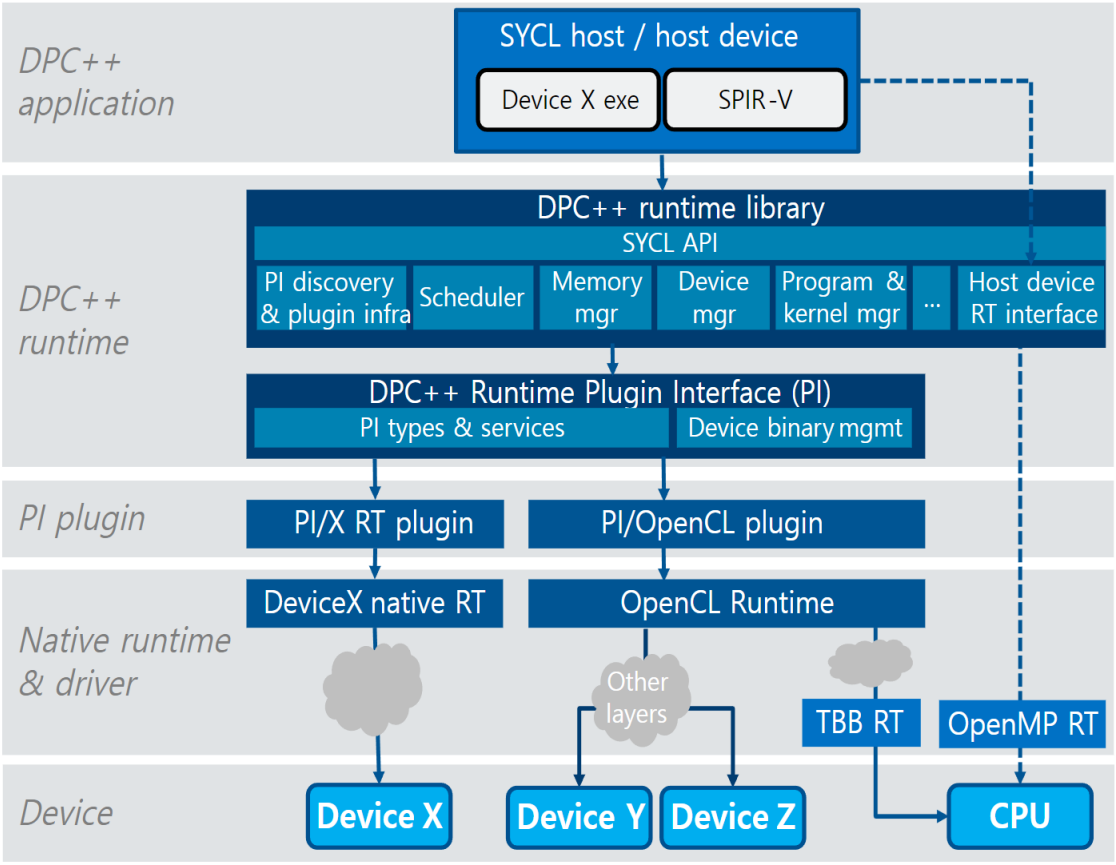

그림 1 Intel DPC++ runtime stack

2. Software stack for accelerator computing

이종컴퓨팅을 위한 소프트웨어 스택은 이종가속기마다, vendor마다 차이가 있을 수 있지만, NVIDIA CUDA를 예시로 보편적인 구조를 소개한다.

CUDA는 크게 Toolkit과 device driver로 구성된다(그림 2). CUDA toolkit에는 runtime, libraries, tools가 포함되고, user-space에 위치한다. Device driver는 크게 user-mode driver와 kernel-mode driver로 나뉜다.

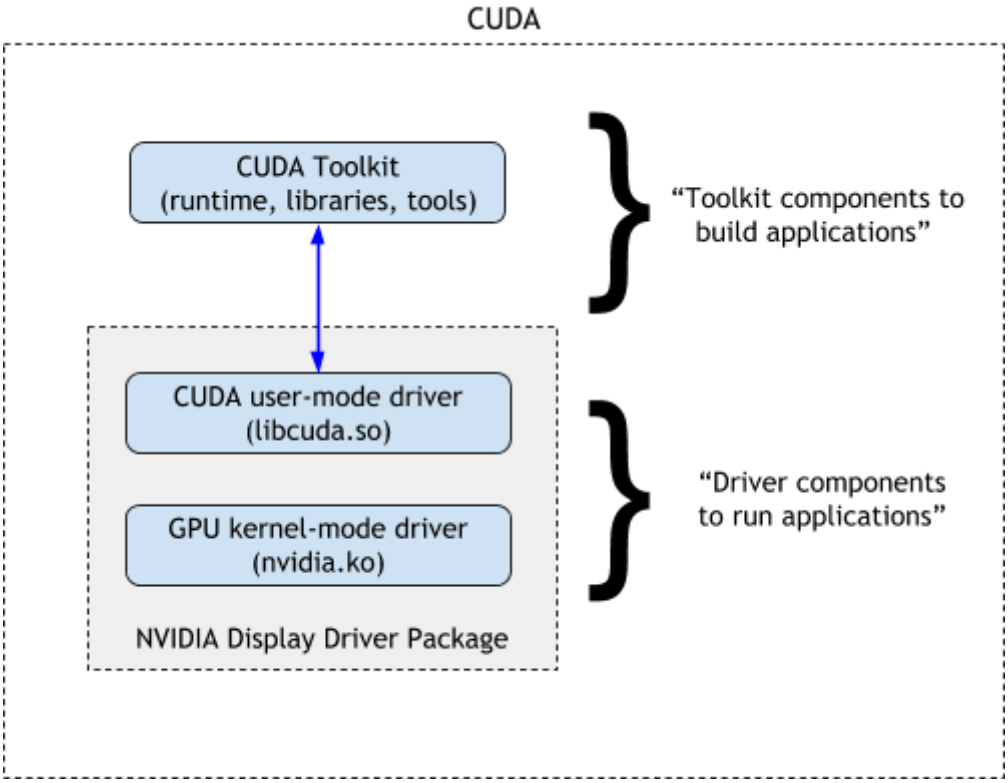

그림 2 Components of CUDA

3. Runtime and device driver for docker container

Docker container는 host의 OS를 공유하기 때문에 이종컴퓨팅 환경을 구축하기 위해서는 user-space 소프트웨어를 별도로 설치해야 한다. User-mode driver 또한 설치해야 한다.

3. Native 환경에서 공용 런타임

1. Intel CPU/FPGA, NVIDIA GPU

Intel Xeon Gold 6230 CPU를 사용하기 위해서, Intel OpenCL 2021.13.11.0.23_160000 runtime을 설치한다.

Intel FPGA PAC D5005를 사용하기 위해서, Intel OneAPI base toolkit을 이용하여 Intel FPGA SDK for OpenCL, pac_s10을 설치한다. 통합된 OneAPI 외에도 FPGA SDK, FPGA OpenCL 등을 별도로 제공하지만, 패키지 관리가 미흡하여 정상적으로 작동하지 않았다.

NVIDIA A100 GPU의 경우에는 CUDA toolkit 11.7과 NVIDIA driver 515.65.07를 설치하여 내장된 OpenCL 3.0 implementation과 Codeplay의 picuda를 이용하여 Intel DPC++ runtime에 연결한다.

각 런타임은 LD_LIBRARY_PATH에 *.so를 지정하고, /etc/OpenCL/vendors/ 디렉토리 아래에 *.icd 파일로 CL runtime의 경로를 지정한다.

그림 3처럼 sycl-ls 명령어를 통해 각 이종가속기의 드라이버와 runtime이 정상적으로 발견되는 것을 볼 수 있다.

```
sihong@worker01:~$ sycl-ls
[opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2021.13.11.0.23_160000]
[opencl:acc:1] Intel(R) FPGA SDK for OpenCL(TM), pac_s10 : Intel PAC Platform (pac_1fd00000) 1.0 [2022.1]
[opencl:cpu:2] Intel(R) OpenCL, Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz 3.0 [2021.13.11.0.23_160000]
[opencl:acc:3] Intel(R) FPGA SDK for OpenCL(TM), pac_s10 : Intel PAC Platform (pac_1fd00000) 1.0 [2022.1]
[ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[ext_oneapi_cuda:gpu:1] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[ext_oneapi_cuda:gpu:2] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[host:host:0] SYCL host platform, SYCL host device 1.2 [1.2]
sihong@worker01:~$
```

그림 3 sycl-ls 명령어 결과

4. Docker container 환경에서 공용 런타임

1. Runtimes

Intel OneAPI base toolkit을 이용하여 Intel CPU, Intel FPGA PAC용 runtime과 library, 특히 FPGA HLS tool을 설치한다. CUDA는 device driver가 포함되지 않은 CUDA toolkit을 이용하여 runtime을 설치하거나, device driver가 함께 포함된 패키지를 사용한다면 kernel-mode driver를 제외하는 옵션을 이용하여 설치한다.

2. User-space device driver

Container는 host OS와 OS module 일부를 공유하고, user-space module은 별도로 설치해야 한다. 따라서 Intel FPGA, NVIDIA GPU를 container 환경에서 사용하기 위해서는 Intel FPGA user-space driver와 NVIDIA user-mode driver를 설치해야한다. NVIDIA의 user-mode driver의 경우, NVIDIA-Linux-[...].run 설치 파일을 --no-kernel-modules 옵션을 이용하여 설치할 수 있다.

반면에, Intel의 경우에는 intel-fpga-addon 패키지를 설치하여 user-space driver를 설치할 수 있는데, aocl install script를 container향으로 수정해야 한다.

i. OS, kernel version checking
ii. sudo keyword 삭제
iii. 설치 실패 시 중간 파일 삭제 기능 disable

Container 내부에서는 기본적으로 root로 동작하기 때문에 sudo keyword가 불필요하며, 일부 패키지 권한 설정 시 문제가 발생하여 설치 실패 routine을 실행하게 된다. 하지만, 실제로는 필요한 user-space modules은 설치가 완료된 상태(client driver 제외)이므로 설치 실패 시 중간 파일을 삭제하는 기능을 disable 해야 한다. Client driver(.fcd)는 'libopae-c.so', 'libMPF.so', 'libintel_opae_mmd.so'의 경로만 지정하면 되는 텍스트 파일로 /opt/Intel/OpenCLFPGA/oneAPI/Boards/ 디렉토리 하위에 device_board_name.fcd(예: intel_s10sx_pac.fcd)로 생성하면 된다.

```
root@a536e3751345:/# sycl-ls
[opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2021.13.11.0.23_160000]
[opencl:acc:1] Intel(R) FPGA SDK for OpenCL(TM), pac_s10 : Intel PAC Platform (pac_1fd00000) 1.0 [2022.1]
[opencl:cpu:2] Intel(R) OpenCL, Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz 3.0 [2021.13.11.0.23_160000]
[ext_oneapi_cuda:gpu:0] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[ext_oneapi_cuda:gpu:1] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[ext_oneapi_cuda:gpu:2] NVIDIA CUDA BACKEND, NVIDIA A100-PCIE-40GB 0.0 [CUDA 11.7]
[host:host:0] SYCL host platform, SYCL host device 1.2 [1.2]
root@a536e3751345:/#
```

그림 4 container 내부에서 sycl-ls 결과

참고문헌

[1] “Intel DPC++ Compiler document”. Online. https://intel.github.io/llvm-docs/GetStartedGuide.html

[2] “SYCL online document”. Online. https://www.khronos.org/sycl/

[3] “DPC++ support for NVIDIA CUDA”. Online. https://intel.github.io/llvm-docs/GetStartedGuide.html#build-dpc-toolchain-with-support-for-nvidia-cuda

[4] “CUDA compatibility”. Online. https://docs.nvidia.com/deploy/cuda-compatibility/index.html

[5] “Docker”. Online. https://www.docker.com/