Last Mile On Democratizing AI

Zhipeng (Howard) Huang, Huawei
Bio

- Principal Engineer in Huawei IT Productline, Standard&Patent Department
- Former project lead for OpenStack cyborg project, co-chair of Kubernetes Policy WG and CNCF SAFE WG, also leading a team currently works on
  - OpenSDS, Open Service Broker API
  - Akraino, SPDK
  - ONNX, Kubeflow
- Father of twin daughters and heavy metal music fan

Slide url: https://goo.gl/QShdUy
Outline

➔ Challenges For AI

➔ Identify The Gap

➔ Introduce Cyborg Project

➔ Model Meets Resource
Everyone is talking about democratizing AI
But it can’t be truly done without an open cloud infrastructure

- Tensorflow, CNTK, Pytorch, Caffe, MXNET, … Basically everything you can find now about major AI related open source projects
- Same goes to majority of the research papers
Define a **Cloud Infrastructure For AI**

- **PaaS**
- **IaaS**
- **Hardware** *(CPU, GPU, FPGA, ASIC, ...)*

**Cloud Infrastructure**

**Heterogeneous Physical Resource**
Outline

➔ Challenges For AI

➔ Identify The Gap

➔ Introduce Cyborg Project

➔ Model Meets Resource
Deep Learning Model Zoo ...

Frameworks:
- Caffe2
- PyTorch
- TensorFlow
- mxnet
- CNTK

Vendor and numeric libraries:
- Apple CoreML
- Nvidia TensorRT
- Intel/Nervana ngraph
- Qualcomm SNPE

O(n^2) pairs
ONNX as the standard intermediate layer

Shared model and operator representation
From $O(n^2)$ to $O(n)$ pairs

Framework backends

Vendor and numeric libraries
Apple CoreML, Nvidia TensorRT, Intel/Nervana ngraph, Qualcomm SNPE, ...
ONNX and Hardware (TensorRT Example)

```bash
$ onnx2trt my_model.onnx -o my_engine.trt
```

Diagram showing the process of converting an ONNX model to a TensorRT engine.
ONNX Help To Bridge, but ...

These are drivers, still software not hardware yet
ONNX Help To Bridge, but in a cloud ...

How can you effectively schedule each one of these onto the right node?
ONNX Help To Bridge, but in a cloud which you have to build ...

How can you effectively manage these nodes that run your framework?
Can we have an AI cloud infrastructure software which
(1) Provides nice abstraction and management of the heterogeneous resources
(2) Is open source and driven by an open community
(3) Facilitates the e2e AI development
Outline

➔ Challenges For AI
➔ Identify The Gap
➔ Introduce Cyborg Project
➔ Model Meets Resource
Cyborg is a general management framework for accelerators

Proud OpenStack Official Project since 2017.09
(https://github.com/openstack/cyborg)
Cyborg Project Overview

- Subteams: release, driver, doc
- Active Chinese Dev wechat group (48 members) from companies like Huawei, China Mobile, Intel, Lenovo, ZTE, Tencent, Nokia, Unionpay, 99Cloud, Xilinx, Inspur, iFlyTech, UC Berkeley, UIUC, CMU
- Lots of gifs...
Cyborg Rocky Release (OpenStack)

Legend
- Pike Finished
- Queens Finished
- Rocky Planned
- Out-of-scope

Python client: cyborg
- cyborg-api
- cyborg-agent
- cyborg-conductor
- cyborg-db
- report

Drivers:
- NVIDIA GPU
- Intel FPGA
- Xilinx FPGA
- SPDK

Storage:
- NVMe SSD

Resources:
- os-acc
- cyborg-db (resource provider)
Cyborg Kubernetes Integration Planning

- Align Cyborg data model with DPI before 1.13 release
- Cyborg DPI Plugin ready when DPI GA
- Consider the possibility of a CRD Acc controller
- Could be utilized by Kubeflow
Outline

➔ Challenges For AI
➔ Identify The Gap
➔ Introduce Cyborg Project
➔ Model Meets Resource
ONNX Provides High Level IR

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ir_version</td>
<td>int64</td>
<td>The ONNX version assumed by the model.</td>
</tr>
<tr>
<td>opset_import</td>
<td>OperatorSetId</td>
<td>A collection of operator set identifiers made available to the model. An</td>
</tr>
<tr>
<td></td>
<td></td>
<td>implementation must support all operators in the set or reject the model.</td>
</tr>
<tr>
<td>producer_name</td>
<td>string</td>
<td>The name of the tool used to generate the model.</td>
</tr>
<tr>
<td>producer_version</td>
<td>string</td>
<td>A string representing the version of the generating tool.</td>
</tr>
<tr>
<td>domain</td>
<td>string</td>
<td>A reverse-DNS name to indicate the model namespace.</td>
</tr>
<tr>
<td>model_version</td>
<td>int64</td>
<td>A version of the model itself, encoded in an integer.</td>
</tr>
<tr>
<td>doc_string</td>
<td>string</td>
<td>A human-readable documentation for this model.</td>
</tr>
<tr>
<td>graph</td>
<td>Graph</td>
<td>The parameterized graph that is evaluated to execute.</td>
</tr>
<tr>
<td>metadata_props</td>
<td>map&lt;string,string&gt;</td>
<td>Named metadata values; keys should be distinct.</td>
</tr>
</tbody>
</table>
### Cyborg Provides Resource Representation

<table>
<thead>
<tr>
<th>name</th>
<th>value</th>
<th>nullable</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs-name</td>
<td>aes-128</td>
<td>False</td>
<td>name of the bitstream (not unique)</td>
</tr>
<tr>
<td>bs-uuid</td>
<td>{uuid}</td>
<td>False</td>
<td>The uuid generated during synthesis</td>
</tr>
<tr>
<td>vendor</td>
<td>Xilinx</td>
<td>False</td>
<td>Vendor of the card</td>
</tr>
<tr>
<td>board</td>
<td>KU115</td>
<td>False</td>
<td>Board type for this bitstream to load</td>
</tr>
<tr>
<td>shell_id</td>
<td>{uuid}</td>
<td>True</td>
<td>Required shell bs-uuid for the bs</td>
</tr>
<tr>
<td>version</td>
<td>1.0</td>
<td>False</td>
<td>Device version number</td>
</tr>
<tr>
<td>driver</td>
<td>SDX</td>
<td>True</td>
<td>Type of driver for this bitstream</td>
</tr>
<tr>
<td>driver_ver</td>
<td>1.0</td>
<td>False</td>
<td>Driver version</td>
</tr>
<tr>
<td>driver_path</td>
<td>/path/</td>
<td>False</td>
<td>Where to retrieve the driver binary</td>
</tr>
<tr>
<td>topology</td>
<td>{CLOB}</td>
<td>False</td>
<td>Function Topology</td>
</tr>
<tr>
<td>description</td>
<td>desc</td>
<td>True</td>
<td>Description</td>
</tr>
<tr>
<td>region_uuid</td>
<td>{uuid}</td>
<td>True</td>
<td>The uuid for target region type</td>
</tr>
<tr>
<td>function_uuid</td>
<td>{uuid}</td>
<td>False</td>
<td>The uuid for bs function type</td>
</tr>
<tr>
<td>function_name</td>
<td>nic-40</td>
<td>True</td>
<td>The function name for this bitstream</td>
</tr>
</tbody>
</table>
When we have model spec (IR) on the one hand, and the resource spec (RR) like what cyborg provides which is gathered from the device reporting, there is still a piece missing when considering the following scenario:

- **Deploy a container cluster** which runs the AI framework (e.g. TF, PyTorch, MXNET,...) to train the model or do the model serving for inference.
- When we do the deployment, the orchestration platform (e.g. kubernetes) which the pipeline (e.g. kubeflow) runs on needs to **schedule the pods to the desired node**.
- For example say we are using the inception v2 ONNX model, we know the depth of the model, the structure of the model, and so many other things. Also say we are using AI ASIC chips that is built inhouse and the capabilities: number of PEs, number of buffers/caches, pipeline structure, are reported to cyborg via the driver.
- But **how could the platform make a good scheduling decision based upon all these information?** Shall it just look at the model spec to make an informative guess? Or shall it just have a hunch at the cyborg device spec?

The answer is to **correlate IR and RR**: model x is best fit for the device y we have in the cloud.
Model Meets Resource: Correlating IR and RR

- Need a new model metadata schema standard to describe the model/IR
- Need a good additive scheduling policy that provides the correlation

IR → Model Metadata Schema → Policy → Scheduler → AI Workload placement decision

RR
Model Meets Resource: Close The Gap

- **PaaS**
  - Kubeflow + Kubernetes + Cyborg

- **IaaS**
  - OpenStack Cyborg

- **Hardware**
  - (CPU, GPU, FPGA, ASIC, ...)

**Shared model and operator representation**
- From $O(n^2)$ to $O(n)$ pairs

**Vendor and numeric libraries**
- Apple CoreML, Nvidia TensorRT, Intel/Neural Network, Qualcomm Snapdragon, Qualcomm Snapdragon 845, ...
Day 2 Keynote Demo

https://www.openstack.org/summit/berlin-2018/
Outline

➔ A few more words: Look Into The Future
Future #1: AI Native Open Infrastructure

Infrastructure As Code - Programmable Framework

Infrastructure As Model - Learnable Framework
Future #1: AI Native Open Infrastructure

Software 1.0: Write a complex fixed scheduler

Where is the scheduler?
Future #1: AI Native Open Infrastructure

Where is the scheduler?

Software 2.0: Write a model that learns the most appropriate scheduling functionality
Future #2: Truly Disruptive AI Technologies

- Causal Model
  - Neural Network (Application)

- Evolution Strategy
  - Hyperparameter Tuning (Application)

- Brain Inspired Circuit
  - Neuromorphic Computing (Infrastructure)
Future #2: Truly Disruptive AI Technologies
Future #3: Making AI Ubiquitous

Full Implementation (API+Sched+DB+Agent+Driver)

Cyborg (Public Cloud)

Cyborg (Private Cloud)

Cyborg (Edge Cloud)

Agent + Driver

Only Driver
Thank You
Backup