RUN_SCENARIO(all-in-one) 日本語

ここではOpenKasugai-controllerプロジェクトに含まれるall-in-one向けのシナリオを実行する手順について記載する。シナリオの実行手順は、クラスタを構築する際に実行したコマンドにより多少異なる。各シナリオの内容、イメージのビルド方法、アーキテクチャについては、以下OpenKasugai-controllerのドキュメントを参照する事。

- https://github.com/openkasugai/controller/tree/main/docs/Demonstrations/jp/OpenKasugai-Demo-for-All-in-One.pdf

前提

- all-in-oneのクラスタ構築が完了していること
- kind-hostから参照可能なローカルなコンテナレジストリ（本ドキュメントでは `harbor` とする）に以下のイメージが登録されていること。ローカルレジストリの設定は、 `Makefile` 内の `DCI_REGISTRY_ADDR_ALT` にて実施する。
  - 四則演算シナリオ
    - `cpufunc_calcapp:1.0.0`
  - 映像推論シナリオ
    - `cpufunc_gst:1.0.0`
    - `gpufunc_dsa:1.0.0`

四則演算シナリオの実行手順

all-in-oneによるクラスタ構築を実施すると、 `./build` ディレクトリが作成される。 このディレクトリには、クラスタ構築時に使用したファイル群が格納される。四則演算シナリオに関連するファイルは、以下のディレクトリ内に配置される。

- `./build/openkasugai-controller/test/sample-data/sample-data-for-all-in-one/calc-func/`

各種リソースの適用

インストール手順にて `リソースを手動登録する場合の手順` を実施した場合、データフローの実行に必要な各種リソースの登録を手動で実施する。データフローの実行に必要なファイルと役割、適用順は以下のとおりである。

適用順	ファイル名	k8sリソース種別	説明
1	cm-cpufunc-config.yaml	ConfigMap	Functionの実行に必要な詳細な設定情報を格納
2	functioninfo.yaml	ConfigMap	Functionを実行するアクセラレータ種別や、Functionがサポートする接続方式に関する情報を格納
3	functiontype.yaml	FunctionType	FunctionのConfigとInfoから、OpenKasugai上で利用可能なFunctionを定義する
4	functionchain.yaml	FunctionChain	Functionの組み合わせを定義する
5	df-cpufunc.yaml	DataFlow	FunctionChainを指定して、DataFlowの配備を行う

データフローの適用

インストール手順にて `一括構築手順` を実施した場合、データフローの適用は、makeコマンドにより実行する

Make command	使用リソース	概要
make dataflow-calcapp-basic	cpu	クラスタIPを用いて四則演算シナリオを実施する
make df-dataflow-calcapp-sriov	cpu, sr-iov NIC	SR-IOVによる仮想NICを用いて四則演算シナリオを実施する
make dataflow-<PatternName>-multi-worker	-	上述のデータフローのファンクションをkind-workerとkind-worker2に分散配置する

データフロー動作確認

四則演算シナリオの動作確認は、データフロー内のデータ送信ファクションとデータ受信ファンクションを用いて実施する。

データ投入

四則演算シナリオのデータ投入は、データ送信ファンクションのPodよりアプリケーションを実行することにより実施する。

```
sudo kubectl exec -it -n cpufunc-calcapp df-calcapp-wbfunction-send-cpu-pod -- /calcapp 1 2 3 4 5
```

実行結果確認

シナリオの実行結果は、受信ファンクションのPodの標準出力を確認することで実施する

```
sudo kubectl logs -n cpufunc-calcapp df-calcapp-wbfunction-rcv-cpu-pod -c cpu-container0
```

以下は標準出力結果の例である。

> `2024-12-10T06:58:35.585Z INFO workspace/main.go:312 start call Do {"req": "inputs:1 inputs:2 inputs:3 inputs:4 inputs:5 results:{operator:\"pluse\" value:15} results:{operator:\"minus\" value:-13} results:{operator:\"multiply\" value:120} results:{operator:\"divide\" value:0.008333333333333333} results:{operator:\"average\" value:30.502083333333335}"}`

映像推論シナリオの実行手順

映像推論シナリオに関連するファイルは以下のディレクトリ内に配置される。

- `./build/openkasugai-controller/test/sample-data/sample-data-for-all-in-one/cpugpu-func`

各種リソースの適用

インストール手順にて `リソースを手動登録する場合の手順` を実施した場合、データフローの実行に必要な各種リソースの登録を手動で実施する。各ファイルの役割とリソースの適用順序は、四則演算シナリオと同様である。

データフローの適用

インストール手順にて `一括構築手順` を実施した場合、データフローの適用は、makeコマンドにより実行する。

Make command	使用リソース	概要

make dataflow-p1c1	cpu	内部NW 接続パターン
make dataflow-p1c2	cpu	外部NW 接続パターン
make dataflow-p2c1	cpu,gpu	GPU連携 内部NW 接続パターン
make dataflow-p2c2	cpu,gpu	GPU連携 内部NW 接続パターン
make dataflow-p2c3	cpu,gpu,sr-iov NIC	GPU連携 外部NW SR-IOV 接続パターン
make dataflow-p2c4	cpu,gpu	GPU連携 内部NW 複数 接続パターン
make dataflow-p3c1	cpu,sr-iov NIC	SR-IOV 接続パターン
make dataflow-<PatternName>-multi-worker	-	上述のデータフローのファンクションを kind-workerとkind-worker2に分散配置する

データフロー動作確認

映像推論データフローへのデータ投入は、映像配信Podからデータフローへ映像データを送信することで実施する。映像配信Podは、クラスタの作成時に自動的に作成される。データフローによる処理結果は、映像受信コンテナのPod内のmp4ファイルを再生することで実施する。

データ投入

映像配信Podから映像を送信するIPアドレスを特定する。データフローにおける先頭ファンクションの`EXTERNAL-IP`を宛先とする。先頭ファンクションは、`functionchain.yaml`を参照して確認する。

```
sudo kubectl get service -n cpufunc-sample df-cpu-pXcY-wbfunction-pip-service
```

```
NAME                                TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)          AGE
df-cpu-pXcY-wbfunction-pip-service   LoadBalancer   10.103.16.154   172.18.9.0      5678:30751/UDP   6m6s
```

映像受信Pod上でbashを実行する。

```
sudo kubectl exec -it -n test send-video-tool -- /bin/bash
```

配信コマンドを実行する。`udpsink host`には、上記で特定した送信先のIPアドレスを指定する。

以下の例では、`./work/DATA/video/`に`sample.mp4`を配置した場合である。コンテナ内で`./work/DATA/video/`は`/opt/video/`としてマウントされる。

```
gst-launch-1.0 -e -v filesrc location=/opt/video/sample.mp4  \
 ! qtdemux  \
 ! video/x-h264  \
 ! h264parse  \
 ! rtph264pay config-interval=-1 seqnum-offset=1  \
 ! udpsink host=172.18.9.0 port=5678 buffer-size=2048
```

実行結果確認

映像受信ファンクションが受信した映像の取得は、以下のコマンドで実行する。

```
sudo kubectl cp -n cpufunc-sample -c cpu-container0 df-cpu-pXcY-wbfunction-rcv-cpu-pod:rcv_video.mp4 rcv_video.mp4
```