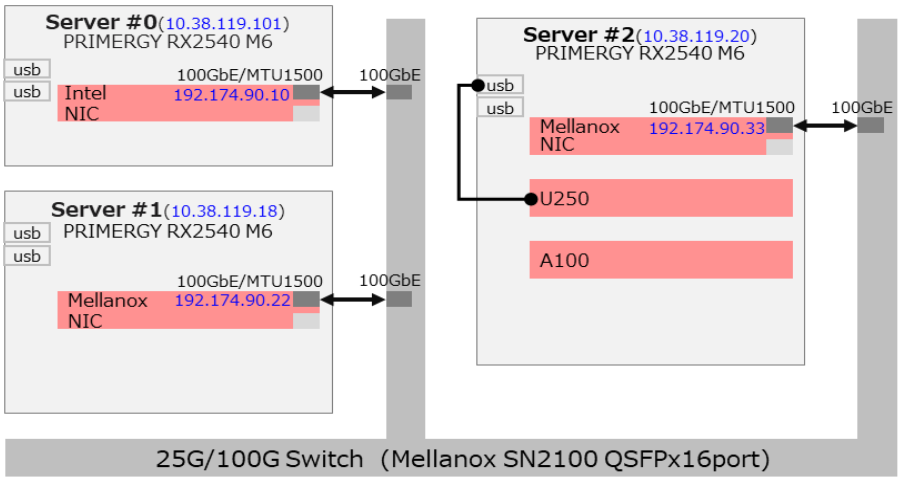
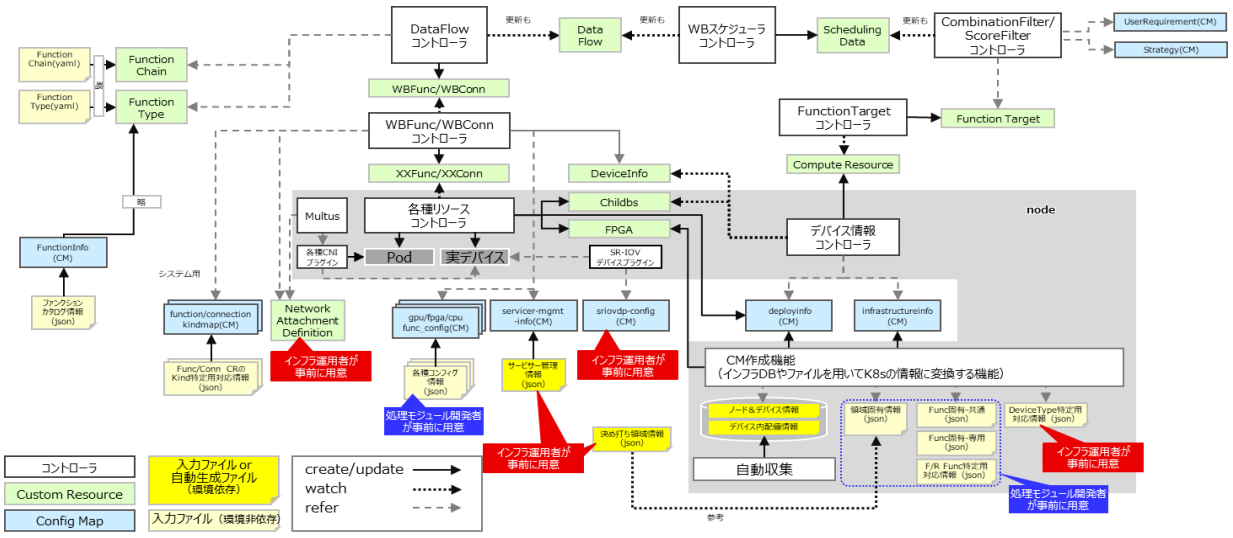
OpenKasugai-Controller
Install Manual(Attachment)

目次	説明
1.想定環境図など	OpenKasugai-Controller Install Manualにて想定している物理構成やソフト構成
2.YAML説明	設定内容についての解説 補足事項シート有り
3.CM作成に使用する入力データ(JSON)の説明	設定内容や設定値についての解説。補足シートも参照のこと 補足事項シート有り
4. CRCのdaemonset用YAML説明	CRCのdaemonset用YAMLの内容についての説明

v1.0.0



物理構成図



全体ソフト構成図

DataFlowを配備するために作成が必要なYAMLファイルについて、~/controller/sample-data/sample-data-demo/yaml/dataflows/test-ext-1/df-test-ext-1-1.yamlを例に説明する
各処理モジュールのPodが使用するネットワーク情報（IPアドレス・ポート番号）の設定方法の詳細については、「2.(補足).DataFlowyamlの設定」を参照のこと

DataFlow YAML	説明	備考
apiVersion: example.com/v1		
kind: DataFlow		
metadata:		
name: "df-test-3-1-1-1"	ユーザが任意に設定する	
namespace: "test01"	ユーザが任意に設定する	
spec:		
functionChainRef:		
name: "cpu-decode-cpu-filter-resize-2types-high-infer-chain"	DataFlowが利用するFunctionChainのmetadata.Name	
namespace: "chain-imgproc"	DataFlowが利用するFunctionChainのmetadata.Namespace	
requirements:	スケジューリング時に満たす必要がある要件を記載する	現在の試作ではファンクションチェイン全体の要件のみが指定可能（全てのファンクションに共通の要件が適用される）
all:	ファンクションチェイン全体の要件を記載	
capacity: 15	想定負荷量(fps)を記載	各コネクションと各コネクションによる想定負荷量(fps)
functionUserParameter:		
- functionKey: decode-main	CPUデコードFunctionの識別子	
userParams:		
ipAddress: 192.174.90.101/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される	SR-IOVのVFを作成した100GNICの物理IPアドレスと同一のサブネットのIPアドレスを指定する
inputPort: 5004	自身のポート番号	
outputIPAddress: 192.174.90.111	送信先（CPUフィルタリサイズ）のIPアドレス	
outputPort: 15000	送信先（CPUフィルタリサイズ）のポート番号	
- functionKey: filter-resize-high-infer-main	CPUフィルタリサイズFunctionの識別子	
userParams:		
ipAddress: 192.174.90.111/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される	SR-IOVのVFを作成した100GNICの物理IPアドレスと同一のサブネットのIPアドレスを指定する
inputPort: 15000	自身のポート番号	
outputIPAddress: 192.174.90.121	送信先（コピー分岐）のIPアドレス	
outputPort: 16000	送信先（コピー分岐）のポート番号	
- functionKey: copy-branch-main	コピー分岐Functionの識別子	
userParams:		
ipAddress: 192.174.90.121/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される	SR-IOVのVFを作成した100GNICの物理IPアドレスと同一のサブネットのIPアドレスを指定する
inputIPAddress: 192.174.90.121	自身のIPアドレス。前段のファンクションとのTCP接続の確立のために利用（サブネットマスクの設定は不要）	自身のipAddressの設定値と同じIPアドレスを設定すれば良い（サブネットマスクの設定は不要）
inputPort: 16000	自身のポート番号	
branchOutputIPAddress: 192.174.90.141,192.174.90.142	送信先（GPU高度推論1、GPU高度推論2）のIPアドレスをカンマ区切りで指定	
branchOutputPort: 17000,18000	送信先（GPU高度推論1、GPU高度推論2）のポート番号をカンマ区切りで指定	
- functionKey: infer-1	GPU高度推論Function1の識別子	
userParams:		
ipAddress: 192.174.90.141/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される	SR-IOVのVFを作成した100GNICの物理IPアドレスと同一のサブネットのIPアドレスを指定する
inputIPAddress: 192.174.90.141	自身のIPアドレス。GStreamerのビデオ処理のコマンド実行(fpsdepay)に利用（サブネットマスクの設定は不要）	自身のipAddressの設定値と同じIPアドレスを設定すれば良い（サブネットマスクの設定は不要）
inputPort: 17000	自身のポート番号	
outputIPAddress: 192.174.90.10	送信先（映像受信ツール）のIPアドレス	
outputPort: 2001	送信先（映像受信ツール）のポート番号	
- functionKey: infer-2	GPU高度推論Function2の識別子	
userParams:		
ipAddress: 192.174.90.142/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される	SR-IOVのVFを作成した100GNICの物理IPアドレスと同一のサブネットのIPアドレスを指定する
inputIPAddress: 192.174.90.142	自身のIPアドレス。GStreamerのビデオ処理のコマンド実行(fpsdepay)に必要	自身のipAddressの設定値と同じIPアドレスを設定すれば良い
inputPort: 18000	自身のポート番号	
outputIPAddress: 192.174.90.10	送信先（映像受信ツール）のIPアドレス	
outputPort: 2002	送信先（映像受信ツール）のポート番号	
userRequirement: user-requirement	DataFlowのスケジューリング用の各種設定情報の取得のために参照するUserRequirementのConfigMapのmetadata.nameを指定	詳細は「OpenKasugai-Controller-InstallManual.j」の「9.7節 DataFlowのスケジューリング戦略を設定する場合」を参照

FunctionChain YAML	説明	備考
apiVersion: example.com/v1		
kind: FunctionChain		
metadata:		
name: cpu-decode-cpu-filter-resize-2types-high-infer-chain	ユーザが任意に設定する	
namespace: chain-imgproc	ユーザが任意に設定する	
spec:		
functionTypeNamespace: "wbfunc-imgproc"	FunctionTypeのNamespace	
connectionTypeNamespace: "default"	ConnectionTypeのNamespace	現在の試作では未使用
functions:	FunctionChainを構成するFunctionのmap keyはConnectionsのFromまたはToで指定するFunctionの識別子（このFunctionChainのリソースにおいてユニークな文字列。各CRのFunctionKeyに入る文字列）	
decode-main:	CPUデコードFuncitonの識別子	
functionName: "cpu-decode"	FunctionTypeSpecで定義したName, Versionを指定	
version: "1.0.0"		
filter-resize-high-infer-main:	CPUフィルタリサイズFunctionの識別子	
functionName: "cpu-filter-resize-high-infer"	FunctionTypeSpecで定義したName, Versionを指定	
version: "1.0.0"		
copy-branch-main:	CPUコピー分岐Functionの識別子	
functionName: "copy-branch"	FunctionTypeSpecで定義したName, Versionを指定	
version: "1.0.0"		
infer-1:	GPU高度推論（1つ目）Functionの識別子	
functionName: "high-infer"	FunctionTypeSpecで定義したName, Versionを指定	
version: "1.0.0"		
infer-2:	GPU高度推論（2つ目）Functionの識別子	
functionName: "high-infer"	FunctionTypeSpecで定義したName, Versionを指定	
version: "1.0.0"		
connections:	FunctionChainを構成するConnectionのリスト	
- from:	Connectionにおける送信元Functionの情報	
functionKey: "wb-start-of-chain"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	・ConnectionのFromがFunctionChain(FC)の開始点(監視カメラなどのデータソース相当)の場合は、"wb-start-of-chain"から始まる文字列を設定すること ・統合ポイントが挿入されたFC等の様に開始点が複数ある場合は、"wb-start-of-chain"の後に数値や文字列を付与して、このFC内でユニークな文字列を設定すること（例１："wb-start-of-chain-1"や"wb-start-of-chain-2"、例2："wb-start-of-chain-xxx"や"wb-start-of-chain-yyy"） ※現状、"wb-start-of-chain"の後に付与された数値や文字列は、制御には用いていない
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "decode-main"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "decode-main"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "filter-resize-high-infer-main"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "filter-resize-high-infer-main"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "copy-branch-main"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "copy-branch-main"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "infer-1"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "copy-branch-main"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 1	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "infer-2"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "infer-1"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "wb-end-of-chain-1"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	・Connectionの1つがFCの終了点(処理結果を受信する外部アプリ相当)の場合は、"wb-end-of-chain"から始まる文字列を設定すること ・分岐ポイントが挿入されたFC等の様に終了点が複数ある場合は、"wb-end-of-chain"の後に数値や文字列を付与して、このFC内でユニークな文字列を設定すること（例１："wb-end-of-chain-1"や"wb-end-of-chain-2"、例2："wb-end-of-chain-xxx"や"wb-end-of-chain-yyy"） ※現状、"wb-end-of-chain"の後に付与された数値や文字列は、制御には用いていない
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない
- from:	Connectionにおける送信元Functionの情報	
functionKey: "infer-2"	データ送信元Functionの識別子。FunctionsのmapのKey値を設定	
port: 0	データ送信元Functionの出力ポート識別番号(Functionが1出力の場合は0を指定)	TCP/UDPのポート番号ではない
to:	Connectionにおける送信先Functionの情報	
functionKey: "wb-end-of-chain-2"	データ送信先Functionの識別子。FunctionsのmapのKey値を設定	・Connectionの1つがFCの終了点(処理結果を受信する外部アプリ相当)の場合は、"wb-end-of-chain"から始まる文字列を設定すること ・分岐ポイントが挿入されたFC等の様に終了点が複数ある場合は、"wb-end-of-chain"の後に数値や文字列を付与して、このFC内でユニークな文字列を設定すること（例１："wb-end-of-chain-1"や"wb-end-of-chain-2"、例2："wb-end-of-chain-xxx"や"wb-end-of-chain-yyy"） ※現状、"wb-end-of-chain"の後に付与された数値や文字列は、制御には用いていない
port: 0	データ送信先Functionの入力ポート識別番号(Functionが1入力の場合は0を指定)	TCP/UDPのポート番号ではない
connectionTypeName: "auto"	ConnectionTypeのリソース名または"auto"を指定	現在の試作は常に"auto"を指定。そのためConnectionTypeカスタムリソースを参照することはない

FunctionType YAML	説明	備考
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-decode	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: decode	ファンクションカタログにおけるFPGAデコードのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-decode	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-cpu-decode	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: cpu-decode	ファンクションカタログにおけるCPUデコードのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-cpu-decode	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-filter-resize-high-infer	ユーザが任意に設定する	

namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: filter-resize-high-infer	ファンクションカタログにおけるGPU高度推論用FPGAフィルタリサイズのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-filter-resize-high-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-filter-resize-low-infer	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: filter-resize-low-infer	ファンクションカタログにおけるGPU軽量推論用FPGAフィルタリサイズのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-filter-resize-low-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-cpu-filter-resize-high-infer	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: cpu-filter-resize-high-infer	ファンクションカタログにおけるGPU高度推論用CPUフィルタリサイズのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-cpu-filter-resize-high-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-cpu-filter-resize-low-infer	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: cpu-filter-resize-low-infer	ファンクションカタログにおけるGPU軽量推論用CPUフィルタリサイズのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-cpu-filter-resize-low-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-copy-branch	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: copy-branch	ファンクションカタログにおけるコピー分岐のファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-copy-branch	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-glue-fdma-to-tcp	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: glue-fdma-to-tcp	ファンクションカタログにおけるGlueのファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-glue-fdma-to-tcp	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-high-infer	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: high-infer	ファンクションカタログにおけるGPU高度推論のファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-high-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	
---		
apiVersion: example.com/v1		
kind: FunctionType		
metadata:		
name: functype-low-infer	ユーザが任意に設定する	
namespace: wbfunc-imgproc	ユーザが任意に設定する	
spec:		
functionName: low-infer	ファンクションカタログにおけるGPU軽量推論のファンクション名 FunctionChainのFunctionNameに設定される値	
functionInfoCMRef:		
name: funcinfo-low-infer	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Nameを指定	
namespace: wbfunc-imgproc	FunctionNameのファンクションが定義されているFunctionInfo(ConfigMap)のmetadata.Namespaceを指定	
version: 1.0.0	Functionのバージョン。Name+Versionで一意性を担保するために使用	

FunctionInfo YAML	**説明**	**備考**
apiVersion: v1		
items:		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-decode	"funcinfo-"+ファンクション名	
namespace: wbfunc-imgproc	ユーザが任意に設定する (ファンクションカタログのカテゴリなどに従い管理者が指定する想定)	
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "alveo",	配備可能な領域種別	
"inputInterfaceType": "dev25gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "dev25gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "fpgafunc-config-decode",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 6,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-cpu-decode		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-decode",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
},		
{		
"name": "item2",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-decode",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	

"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 20,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-filter-resize-high-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "alveo",	配備可能な領域種別	
"inputInterfaceType": "dev25gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "fpgafunc-config-filter-resize-high-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
},		
{		
"name": "item2",	deployableItems配列の要素を参照するための名前	
"regionType": "alveo",	配備可能な領域種別	
"inputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "fpgafunc-config-filter-resize-high-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 8,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 40,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-filter-resize-low-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "alveo",	配備可能な領域種別	
"inputInterfaceType": "dev25gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "fpgafunc-config-filter-resize-low-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
},		
{		
"name": "item2",	deployableItems配列の要素を参照するための名前	
"regionType": "alveo",	配備可能な領域種別	
"inputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "fpgafunc-config-filter-resize-low-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 8,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 40,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-cpu-filter-resize-high-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-filter-resize-high-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-cpu-filter-resize-low-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-filter-resize-low-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		

]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-copy-branch		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-copy-branch",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":10	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-glue-fdma-to-tcp		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "cpu",	配備可能な領域種別	
"inputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "cpufunc-config-glue-fdma-to-tcp",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-high-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "a100",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "gpufunc-config-high-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
},		
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "a100",	配備可能な領域種別	
"inputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "gpufunc-config-high-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 15,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
- apiVersion: v1		
kind: ConfigMap		
metadata:		
name: funcinfo-low-infer		
namespace: wbfunc-imgproc		
data:		
deployableItems: '[	以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "t4",	配備可能な領域種別	
"inputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "gpufunc-config-low-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
},		
{		
"name": "item1",	deployableItems配列の要素を参照するための名前	
"regionType": "t4",	配備可能な領域種別	
"inputInterfaceType": "mem",	上記の<regionType>に配備した場合に使用可能な入力のインターフェース種別	
"outputInterfaceType": "host100gether",	上記の<regionType>に配備した場合に使用可能な出力のインターフェース種別	
"configName": "gpufunc-config-low-infer",	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合に、デプロイに必要な情報の名前	
"specName": "spec1"	上記の<regionType>に配備して上記の<inputInterfaceType>と<outputInterfaceType>を使用する場合のファンクションのスペック情報の名前	
}		
]'		
spec: '[	ファンクションのスペック情報 以下key-valueからなるjsonオブジェクトを要素とする配列を文字列にした値	現在の試作では未使用
{		
"name": "spec1",	spec配列の要素を参照するための名前	
"minCore": 1,	使用するリソースの最小値	
"maxCore": 1,	使用するリソースの最大値	
"maxDataFlowsBase": 1,	基本の最大割合DataFlow数(最大搭載WBFunction数)。回路のチャネル数等によって決まる	
"maxCapacityBase": 5,	基本の最大処理能力(fps)	
"maxInputNum": 1,	ファンクションの最大入力数	
"maxOutputNum":1	ファンクションの最大出力数	
}		
]'		
kind: List		

「2.YAML説明」のDataFlowのYAMLファイルの設定内容のうち、各処理モジュールのPodが使用するネットワーク情報（IPアドレス・ポート番号）の設定について説明する
以下のAに全処理モジュール共通の設定を、以下のBに各処理モジュール毎の設定内容を記載する

DataFlow YAML	説明
apiVersion: example.com/v1	
kind: DataFlow	
metadata:	
name: "df-test-3-1-1-1"	ユーザが任意に設定する
namespace: "test01"	ユーザが任意に設定する
spec:	
functionChainRef:	
name: "cpu-decode-cpu-filter-resize-2types-high-infer-chain"	DataFlowが利用するFunctionChainのmetadata.Name
namespace: "chain-imgproc"	DataFlowが利用するFunctionChainのmetadata.Namespace
requirements:	スケジューリング時に満たす必要がある要件を記載する
all:	ファンクションチェイン全体の要件を記載
capacity: 15	想定負荷量(fps)を記載
functionUserParameter:	
- functionKey: decode-main	CPUデコードFunctionの識別子
userParams:	
ipAddress: 192.174.90.101/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
inputPort: 5004	自身のポート番号
outputIPAddress: 192.174.90.111	送信先（CPUフィルタリサイズ）のIPアドレス
outputPort: 15000	送信先（CPUフィルタリサイズ）のポート番号
- functionKey: filter-resize-high-infer-main	CPUフィルタリサイズFunctionの識別子
userParams:	
ipAddress: 192.174.90.111/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
inputPort: 15000	自身のポート番号
outputIPAddress: 192.174.90.121	送信先（コピー分岐）のIPアドレス
outputPort: 16000	送信先（コピー分岐）のポート番号
- functionKey: copy-branch-main	コピー分岐Functionの識別子
userParams:	
ipAddress: 192.174.90.121/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
inputIPAddress: 192.174.90.121	自身のIPアドレス。前段のファンクションとのTCP接続の確立のために必要
inputPort: 16000	自身のポート番号
branchOutputIPAddress: 192.174.90.141,192.174.90.142	送信先（GPU高度推論1、GPU高度推論2）のIPアドレスをカンマ区切りで指定
branchOutputPort: 17000,18000	送信先（GPU高度推論1、GPU高度推論2）のポート番号をカンマ区切りで指定
- functionKey: infer-1	GPU高度推論Function1の識別子
userParams:	
ipAddress: 192.174.90.141/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
inputIPAddress: 192.174.90.141	自身のIPアドレス。GStreamerのビデオ処理のコマンド実行(fpgdepay)に必要
inputPort: 17000	自身のポート番号
outputIPAddress: 192.174.90.10	送信先（映像受信ツール）のIPアドレス
outputPort: 2001	送信先（映像受信ツール）のポート番号
- functionKey: infer-2	GPU高度推論Function2の識別子
userParams:	
ipAddress: 192.174.90.142/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
inputIPAddress: 192.174.90.142	自身のIPアドレス。GStreamerのビデオ処理のコマンド実行(fpgdepay)に必要
inputPort: 18000	自身のポート番号
outputIPAddress: 192.174.90.10	送信先（映像受信ツール）のIPアドレス
outputPort: 2002	送信先（映像受信ツール）のポート番号
userRequirement: user-requirement	DataFlowのスケジューリング用の各種設定情報の取得のために参照するUserRequirementのConfigMapのmetadata.nameを指定

A.全処理モジュール共通の設定内容について

・自身がPod上で動作する処理モジュールであり、
自身に対する送信元が別のPod上で動作する処理モジュールまたは映像配信ツールの場合は、ipAddressとinputPortを設定する
・自身がPod上で動作する処理モジュールであり、
自身からの送信先が別のPodで動作する処理モジュールまたは映像受信ツールの場合は、ipAddressとoutputIPAddressとoutputPortを設定する
・一部の処理モジュールでは、当該処理モジュールの固有の処理のために、inputIPAddressを設定する
・ipAddress、inputIPAddress、outputIPAddressを設定する場合、
「OpenKasugai-Controller-InstallManual」の「8.7 SR-IOVのVF作成および管理」で
VFを作成した100GNICの物理IPアドレスと共通のサブネットのIPアドレスを設定する
・Pod上で動作しないFPGAデコード/FPGAフィルタリサイズの処理モジュールの場合は、上記いずれの設定も不要

B.各処理モジュール毎の設定内容について

A.CPUデコードを使用する場合

functionKey: decode-main
userParams:
① ipAddress: 192.174.90.101/24　　自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
② inputPort: 5004　　自身のポート番号
③ outputIPAddress: 192.174.90.111　　送信先のIPアドレス
④ outputPort: 15000　　送信先のIPアドレス

②についての補足：ポート番号の値は映像配信ツールで指定したCPUデコードアプリのポート番号を記載する。
以下はOpenKasugai-Demoの1.3.3節(1)の①を抜粋

```
./start_gst_sender.sh        /opt/video/pocdemo_movie/day_scene/d1_12_Videvo-4_2K_160929_057_London_WestminsterBridge7_1080p_5min_conv_4K_8Mbps_15fps.mp4 192.174.90.101 5004 1 ${sleep_time:-3}↵
↵
./start_gst_sender.sh        /opt/video/pocdemo_movie/day_scene/d2_06_Pexels-15_4K_pexels-creativ-medium-5607960_5min_conv_4K_6Mbps_15fps.mp4 192.174.90.102 5004 1 ${sleep_time:-3}↵
```

B.CPUフィルタリサイズを使用する場合

functionKey: filter-resize-xxx-infer-main　　高度推論用の場合は filter-resize-high-infer-mainを、軽量推論用の場合は filter-resize-low-infer-mainをfunctionKeyとして指定
userParams:
① ipAddress: 192.174.90.111/24　　自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
② inputPort: 15000　　自身のポート番号
③ outputIPAddress: 192.174.90.121　　送信先のIPアドレス
④ outputPort: 16000　　送信先のポート番号

C.コピー分岐を使用する場合

functionKey: copy-branch-main
userParams:

①	ipAddress: 192.174.90.121/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
②	inputIPAddress: 192.174.90.121	自身のIPアドレス。前段の処理モジュールとのTCP接続の確立のために利用（サブネットマスクの設定は不要）
③	inputPort: 16000	自身のポート番号
④	branchOutputIPAddress: 192.174.90.141,192.174.90.142	送信先のIPアドレスをカンマ区切りで指定
⑤	branchOutputPort: 17000,18000	送信先のポート番号をカンマ区切りで指定

②についての補足：①と同じIPアドレスを設定する(サブネットマスクの記載は不要)
④についての補足：送信先の分岐数分、IPアドレスをカンマ区切りで設定する
⑤についての補足：送信先の分岐数分、ポート番号をカンマ区切りで設定する

D.GPU推論を使用する場合

functionKey: xxx-infer-main または infer-[n]　DataFlowにコピー分岐なしの場合、高度推論の場合はhigh-infer-mainを、軽量推論の場合はlow-infer-mainをfunctionKeyとして指定
userParams:　DataFlowにコピー分岐ありの場合、分岐した内の何個目のGPU推論であるかに応じて、infer1またはinfer2をfunctionKeyとして指定

①	ipAddress: 192.174.90.141/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
②	inputIPAddress: 192.174.90.141	自身のIPアドレス。GStreamerのビデオ処理のコマンド実行(fpgdepay)に利用（サブネットマスクの設定は不要）
③	inputPort: 16000	自身のポート番号
④	outputIPAddress: 192.174.90.10	映像受信ツールのIPアドレス
⑤	outputPort: 2001	映像受信ツールのポート番号

②についての補足：①と同じIPアドレスを設定する(サブネットマスクの記載は不要)
④についての補足：IPアドレスの値は映像受信ツールが使用するK8s control planeの100GNICのIPアドレスを指定する
K8s control planeで以下のコマンドを実行し、IPアドレスを確認する。

```
$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether b4:96:91:9d:79:80 brd ff:ff:ff:ff:ff:ff
    inet 10.38.119.15/24 brd 10.38.119.255 scope global ens1f0
       valid_lft forever preferred_lft forever
    inet6 fe80::b696:91ff:fe9d:7980/64 scope link
       valid_lft forever preferred_lft forever

~省略~

7: ens3f1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9000 qdisc mq state UP group default qlen 1000
    link/ether 0c:42:a1:6d:65:35 brd ff:ff:ff:ff:ff:ff
    inet 192.174.90.10/24 brd 192.174.91.255 scope global ens3f1
       valid_lft forever preferred_lft forever
    inet6 fe80::e42:a1ff:fe6d:6535/64 scope link
       valid_lft forever preferred_lft forever
```

⑤についての補足：ポート番号の値は評価手順の映像受信開始手順で指定したポート番号を記載する。
sample-demoの例：2001か2002を指定する。
以下はOpenKasugai-Demoの1.3.2節の(1)の①を抜粋
(1) 映像受信の起動方法
① 映像受信ツールの配備と受信用スクリプト（高度推論結果受信×2）の作成
（初回のみ実施、2回目以降は手順①は不要）

```
$ cd ~/openkasugai-controller/sample-functions/utils/rcv_vide_tool/
$ kubectl create ns test
$ kubectl apply -f rcv_video_tool.yaml
$ kubectl exec -n test -it rcv-video-tool-XXX -- bash  ※pod名は環境に合わせて変更
# vi start_test1.sh  ※以下を貼り付けで保存する
#!/bin/bash -x

for i in `seq -w 01 02`
do
    gst-launch-1.0 -e udpsrc buffer-size=21299100 mtu=8900 port=20${i} ! 'application/x-rtp, media=(string)video, clock-rate=(int)90000, encoding-name=(string)RAW, sampling=(string)BGR, depth=(string)8, width=(string)1280, height=(string)1280, payload=(int)96' ! rtpvrawdepay ! queue ! videoconvert ! 'video/x-raw, format=(string)I420' ! openh264enc ! 'video/x-h264, stream-format=byte-stream, profile=(string)high' ! perf name=stream${i} ! h264parse ! qtmux ! filesink location=/tmp/output_st${i}.mp4 sync=false > /tmp/rcv_video_tool_st${i}.log &
done

# chmod +x start_test1.sh
# exit
```

E.Glueを使用する場合 ※2.YAML説明のDataFlowでは利用していない処理モジュール

functionKey: glue-fdma-to-tcp-main
userParams:

①	ipAddress: 192.174.90.131/24	自身のIPアドレス。Podの2nd NICのIPアドレスとして設定される
②	glueOutputIPAddress: 192.174.90.141	送信先のIPアドレス
③	glueOutputPort: 16000	送信先のポート番号

F.FPGAデコード/FPGAフィルタリサイズを使用する場合

設定不要

B. 提供した資材から編集が必要なファイル

決め打ち領域情報

※k8sクラスタ内で使用する全領域に関して、各領域の領域種別を記載する必要あり。

predetermined-region.json	説明	環境に合わせた変更が必要※	備考(どの値を記載すれば良いかなど)
[			領域単位で設定する
{	1領域分の情報		
"nodeName": "worker0",	その領域があるノードのノード名	○	
"deviceUUID": "21330621T01J",	その領域があるデバイスの識別情報	○	・FPGAの場合："ls /dev/*"コマンドの結果を利用。FPGAdデバイスは"xpcie_${FPGA-ID}"と表示されるので、${FPGA-ID}の値を記載する ・GPUの場合："nvidia-smi -L"コマンドの結果を利用。デバイス毎にUUID(例：GPU-b8b4f1f5-bf51-eaa3-6ec4-97190b7f6c98)が出力されるのでその値を記載する ・CPUの場合："0"を記載する（現状各サーバで仮想的に１つとみなしているため、固定値で良い）
"subDeviceSpecRef": "lane0",	その領域の領域名	○	・FPGAの場合："lane"+"${lane番号}"を記載する ・CPU/GPUの場合：deviceTypeと同じ値を記載する
"regionType": "alveou250-0100001c-2lanes-0nics"	その領域の領域種別	○	・FPGAの場合：以下のフォーマットで記載："${デバイス種別}" + "-" + "${親bs-id}" + "-" + "${lane数}" + "lanes" + "-" + "${NIC数}" + "nics"」 ・サンプルとして用意しているFPGA回路(OpenKasugai-Controller-InstallManualの0.4節に記載)を用いる場合は、 デバイス種別="alveou250", 親bs-id="0100001c", lane数="2", NIC数="0"で固定なので、 regionTypeは必ず"alveou250-0100001c-2lanes-0nics"となる。 ・CPU/GPUの場合：deviceTypeと同じ値を記載する
},			
{	1領域分の情報		
"nodeName": "worker0",		○	
"deviceUUID": "21330621T01J",		○	
"subDeviceSpecRef": "lane1",		○	
"regionType": "alveou250-0100001c-2lanes-0nics"		○	
},			
…			
{	1領域分の情報		
"nodeName": "worker1",		○	
"deviceUUID": "21330621T00Y",		○	
"subDeviceSpecRef": "lane0",		○	
"regionType": "alveou250-0100001c-2lanes-0nics"		○	
},			
{	1領域分の情報		
"nodeName": "worker1",		○	
"deviceUUID": "21330621T00Y",		○	
"subDeviceSpecRef": "lane1",		○	
"regionType": "alveou250-0100001c-2lanes-0nics"		○	
},			
{	1領域分の情報		
"nodeName": "worker1",		○	
"deviceUUID": "gpu-702fb653-43a4-732d-6bc4-7b3487696c90"		○	
"subDeviceSpecRef": "a100",		○	
"regionType": "a100"		○	GPU/CPUの領域種別の値はデバイス種別(deviceType)と同等
}			
]			

A. 全てのDFで共通的に使用可能なファイル（環境に合わせた変更が不要でそのまま使える）

デバイスタイプマッピング情報

自動取得したデバイスのモデル名からDeviceTypeに変換するためのマッピング情報。

※サンプルデータには以下の6種類のデバイス(Alveo U250, NVIDIA GPU T4, NVIDIA GPU A100, Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz, Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz, Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz)について記載している。それ以外のデバイスを使用する場合は追記が必要。

devicetypemap.json	説明	備考
[{	1デバイス分の情報	
"inputDeviceType": "ALVEO U250 PQ",	インフラから取得出来るデバイス情報	「自動収集&CM作成ツール用事前情報チェックツール(DeviceInfoCheck.sh)を実行して得られた値を設定している
"outputDeviceType": "alveou250"	上記デバイスに対応するDeviceType	"Alveo系"は"alveo"決め打ち
},{	1デバイス分の情報	
"inputDeviceType": "Tesla T4",		「自動収集&CM作成ツール用事前情報チェックツール(DeviceInfoCheck.sh)を実行して得られた値を設定している
"outputDeviceType": "t4"		"Tesla T4"は"t4"決め打ち
},{	1デバイス分の情報	
"inputDeviceType": "NVIDIA A100 80GB PCIe",		「自動収集&CM作成ツール用事前情報チェックツール(DeviceInfoCheck.sh)を実行して得られた値を設定している
"outputDeviceType": "a100"		"NVIDIA A100 80GB PCIe"は"a100"決め打ち
},{	1デバイス分の情報	
"inputDeviceType": "Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz",		grep model.name /proc/cpuinfo | sort -u の結果を確認し、設定
"outputDeviceType": "cpu"		"Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz"は"cpu"決め打ち
},{	1デバイス分の情報	
"inputDeviceType": "Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz",		grep model.name /proc/cpuinfo | sort -u の結果を確認し、設定
"outputDeviceType": "cpu"		"Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz"は"cpu"決め打ち
},{	1デバイス分の情報	
"inputDeviceType": "Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz",		grep model.name /proc/cpuinfo | sort -u の結果を確認し、設定
"outputDeviceType": "cpu"		"Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz"は"cpu"決め打ち
}]		

領域固有情報
領域毎の性能(配備容量と最大処理能力)を記載する。

・領域の性能は、FPGAならばデバイス種別(デバイスのモデル)と書込む子bitstreamによって変わる想定のため使用するデバイス種別×子bitstream分の記載が必要。CPU/GPUの領域は現状はデバイス種別のみによって変わる想定のため使用するデバイス種別分の記載が必要。
・サンプルデータには以下の6種類の領域について記載している。それ以外の領域を用いる場合は追記が必要。
・FPGAについては1種類の領域について記載。具体的には、Alveo U250にフィルタリサイズ用のbitstreamが書込まれている領域について記載
・GPUについては2種類の領域について記載。具体的には、2種類のデバイス(NVIDIA GPU T4, NVIDIA GPU A100)用の領域について記載
・CPUについては3種類の領域について記載。具体的には3種類のデバイス(Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz と Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz と Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz)用の領域について記載

region-unique-info.json	説明	備考
[{	1デバイス分の領域情報	Alveo U250にフィルタリサイズ用子bitstreamが書込まれている領域分
"subDeviceSpecRef": "0100001c",	デバイスに書込むオブジェクトのId	FPGAではそのデバイスに書込む子bitstreamのId(ここではフィルタリサイズ用bitstreamのId)を記載。
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	lane0分
"regionName": "lane0",	当該領域の領域名(デバイス内で一意)	サンプル子bitstream(フィルタリサイズ用子bitstream)を使用する場合は"lane0"もしくは"lane1"決め打ち
"regionType": "alveou250-0100001c-2lanes-0nics",	当該領域の領域種別	FPGAの領域種別の値は以下のフォーマットで記載：["${デバイス種別}" + "-" + "${親bs-id}" + "-" + "${lane数}" + "lanes" + "-" + "${NIC数}" + "nics"]
"maxFunctions": 1,	当該領域でのFunction(回路/Pod)を配備容量	当該領域に搭載可能な最大Function(回路/Pod)数
"maxCapacity": 40	当該領域の最大処理能力	単位はfps。値は暫定値。
},{		lane1分
"regionName": "lane1",		
"regionType": "alveou250-0100001c-2lanes-0nics",		
"maxFunctions": 1,		
"maxCapacity": 40		
}]		
},{	1デバイス分の領域情報	NVIDIA GPU T4用の領域分
"subDeviceSpecRef": "Tesla T4",	デバイスに書込むオブジェクトのId	GPUとCPUではデバイス種別(デバイスのモデル)名を記載
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	GPUはデバイス毎に1領域(デバイス全体を1つの領域)
"regionName": "t4",		"Tesla T4"は"t4"決め打ち
"regionType": "t4",		CPUとGPUではデバイス種別(deviceType)と同じ値
"maxFunctions": 110,		当該領域に搭載可能な最大Function(回路/Pod)数。値は暫定値。
"maxCapacity": 40		単位はfps。値は暫定値。
}]		
},{	1デバイス分の領域情報	NVIDIA GPU A100用の領域分
"subDeviceSpecRef": "NVIDIA A100 80GB PCIe",	デバイスに書込むオブジェクトのId	GPUとCPUではデバイス種別(デバイスのモデル)名を記載
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	GPUはデバイス毎に1領域(デバイス全体を1つの領域)
"regionName": "a100",		"NVIDIA A100 80GB PCIe"は"a100"決め打ち
"regionType": "a100",		CPUとGPUではデバイス種別(deviceType)と同じ値
"maxFunctions": 110,		当該領域に搭載可能な最大Function(回路/Pod)数。値は暫定値。
"maxCapacity": 120		単位はfps。値は暫定値。
}]		
},{	1デバイス分の領域情報	Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz用の領域分
"subDeviceSpecRef": "Intel(R) Xeon(R) Gold 6346 CPU @ 3.10GHz",	デバイスに書込むオブジェクトのId	GPUとCPUではデバイス種別(デバイスのモデル)名を記載
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	CPUはデバイス毎に1領域(デバイス全体を1つの領域)
"regionName": "cpu",		CPUは"cpu"決め打ち
"regionType": "cpu",		CPUとGPUではデバイス種別(deviceType)と同じ値
"maxFunctions": 110,		当該領域に搭載可能な最大Function(回路/Pod)数。値は暫定値。
"maxCapacity": 120		単位はfps。値は暫定値。
}]		
},{	1デバイス分の領域情報	Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz用の領域分
"subDeviceSpecRef": "Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz",	デバイスに書込むオブジェクトのId	GPUとCPUではデバイス種別(デバイスのモデル)名を記載
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	CPUはデバイス毎に1領域(デバイス全体を1つの領域)
"regionName": "cpu",		CPUは"cpu"決め打ち
"regionType": "cpu",		CPUとGPUではデバイス種別(deviceType)と同じ値
"maxFunctions": 110,		当該領域に搭載可能な最大Function(回路/Pod)数。値は暫定値。
"maxCapacity": 120		単位はfps。値は暫定値。
}]		
},{	1デバイス分の領域情報	Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz用の領域分
"subDeviceSpecRef": "Intel(R) Xeon(R) Gold 6330 CPU @ 2.00GHz",	デバイスに書込むオブジェクトのId	GPUとCPUではデバイス種別(デバイスのモデル)名を記載
"functionTargets":[{	このオブジェクトを書込んだ場合に作られる領域の数だけ記載する	CPUはデバイス毎に1領域(デバイス全体を1つの領域)
"regionName": "cpu",		CPUは"cpu"決め打ち
"regionType": "cpu",		CPUとGPUではデバイス種別(deviceType)と同じ値
"maxFunctions": 110,		当該領域に搭載可能な最大Function(回路/Pod)数。値は暫定値。
"maxCapacity": 120		単位はfps。値は暫定値。
}]		
}]		

FunctionType特定用対応情報
使用するDeviceKindの数分のエントリを記載する
※サンプルデータには4種類のデバイスタイプ(alveou250, t4, a100, cpu)について記載している。それ以外のデバイスタイプを使用する場合は、デバイスタイプマッピング情報に追加デバイスタイプ分の情報を追記するとともに、本情報に追加デバイスタイプ分の情報の追記が必要。
追加デバイス分の領域に関する情報の追記が必要。追記を行う場合の追記方法は備考欄を参照のこと。

functionkindmap.json	説明	備考
[{	1種類のFunction系CR分の情報	FPGAFunction用の情報
"deviceType": "alveou250",	key: WBFuncでのDeviceTypeの文字列	該当するDeviceType
"functionCRKind":"FPGAFunction"	value: Function系CRの種類	deviceType="alveou250"ならFPGAなので"FPGAFunction"決め打ち
},{	1種類のFunction系CR分の情報	GPUFunction用の情報
"deviceType": "t4",		該当するDeviceType(NVIDIA GPU T4向け)。
"functionCRKind":"GPUFunction"		deviceType="t4"ならGPUなので"GPUFunction"決め打ち
},{	1種類のFunction系CR分の情報	GPUFunction用の情報
"deviceType": "a100",		該当するDeviceType(NVIDIA GPU A100向け)。
"functionCRKind":"GPUFunction"		deviceType="a100"ならGPUなので"GPUFunction"決め打ち
},{	1種類のFunction系CR分の情報	CPUFunction用の情報
"deviceType": "cpu",		該当するDeviceType(各種cpu向け)。
"functionCRKind":"CPUFunction"		deviceType="cpu"ならCPUなので"CPUFunction"決め打ち
}]		

ConnectionType特定用対応情報
使用するコネクションのType数分のエントリを記載する
※サンプルデータには2種類の接続種別(共有メモリ経由のPCIe接続とEthernet接続)について記載している。それ以外の接続種別を使う場合は追記が必要。

connectionkindmap.json	説明	備考
[{	1種類のConnection系CR分の情報	PCIeConnection用の情報
"connectionMethod": "host-mem",	key: WBConnectionでのconnectionMethodの文字列	共有メモリ経由のPCIe接続を行うConnectionMethod
"connectionCRKind": "PCIeConnection"	value: Connection系CRの種類	connectionMethod="host-mem"ならPCIe接続なので"PCIeConnetion"決め打ち
},{	1種類のConnection系CR分の情報	EthernetConnection用の情報
"connectionMethod": "host-100gether",		Ethernet接続を行うConnectionMethod
"connectionCRKind": "EthernetConnection"		connectionMethod="host-100gether"ならEther接続なので"EthernetConnetion"決め打ち
}]		

Function固有情報 - 共通属性
※FPGAのファンクション種別毎にエントリを記載する。 ※GPUのファンクションについては現状記載不要(将来的に記載する方向に変更の可能性はあり)。
※サンプルデータには"filter-resize"についてのみ記載している。それ以外のFPGAのファンクションを追加する場合は追記が必要。また、既存のファンクションについても性能値が変化した場合は更新が必要。

function-unique-info.json	説明	備考
[{	1Function分の情報	filter/resizeのFuntion用の情報
"functionID" : 0,	当該Function(回路/コンテナイメージ)の識別子	現状使っていないので値は空でも良い。
"functionName" : "filter-resize",	当該Function名	
"maxDataFlows": 8,	当該Functionに配備可能な最大DF(WBFunc)数	FPGA回路の場合は同時に提供可能なFunctionChannelID数を記載。
"maxCapacity": 40	当該Functionの最大処理能力	
}]		

Function固有情報 - filter/resize専用属性
※FunctionChannelIDListは、filter/resize(F/R)用のビットストリームで書込んだ全filter/resizeファンクション(1FPGAデバイス毎に2ファンクション分)で提供するFunctionChannelIDの数だけエントリを記載する。
※"Rx"と"Tx"に関して、送信元もしくは送信先との接続タイプが複数考えられる場合はそれぞれ用の値の定義が必要。

filter-resize-childbs.json	説明	備考
{	filter/resize用bitstreamの1lane分のリソース情報	
"functionKernels":[{	1laneに配備されるFunction1つ分で用意するリソースプールの情報	filter/resize用bitstreamの場合は1laneにつき1Functionのみ
"partitionName": "0",	当該FunctionのId	filter/resizeの場合1laneにつき1Functionなので、ここの値はlaneのId(0か1)になる。
"functionChannelIDList": [0,1,2,3,4,5,6,7],	当該Functionで提供するFunctionChannelIDのリスト	当該Functionで提供するFunctionChannelIDを全て記載する。filter/resizeの場合は[0,1,2,3,4,5,6,7]固定で良い
"functionChannelIDs":[{	1FunctionChannelID分のリソース情報	各FPGAFunctionに割り振られる各FunctionChannelID用に用意されたリソース情報。
"functionChannelID": 0,	対象となるFunctionChannelID	上記functionChannelIDList内の値になる。
"rx":{	当該FunctionChannelID向けに提供する受信用リソース群	
"protocol":{	通信可能なプロトコル毎に記載する	
"TCP":{	送信元との通信のプロトコルがTCPの場合に割り振るリソース群	
"port": 12300	送信元との通信で使用するポート番号	
},		
"DMA":{	送信元との通信のプロトコルがDMAの場合に割り振るリソース群	
"port": 12300,	送信元との通信で使用するポート番号	
"lldmaConnectorID": 0,	送信元とDMA通信する際に使用するLLDMAのコネクタId	
"dmaChannelID": 0	送信元とDMA通信する際に使用するDMAのチャネルId	
}		
}		
},		
"tx":{	上記FunctionChannelID向けに提供する送信用リソース群	
"protocol":{	通信可能なプロトコル毎に記載する	
"TCP":{	宛先との通信のプロトコルがTCPの場合に割り振るリソース群	
"port": 12300	宛先との通信で使用するポート番号	
},		
"DMA":{	宛先との通信のプロトコルがDMAの場合に割り振るリソース群	
"port": 12300,	宛先との通信で使用するポート番号	
"lldmaConnectorID": 0,	宛先とDMA通信する際に使用するLLDMAのコネクタId	
"dmaChannelID": 0	宛先とDMA通信する際に使用するDMAのチャネルId	
}		
}		
}		
},{	1FunctionChannelID分のリソース情報	
"functionChannelID": 1,	対象となるFunctionChannelID	
"rx":{	当該FunctionChannelID向けに提供する受信用リソース群	
...		
},		
"tx":{	当該FunctionChannelID向けに提供する送信用リソース群	
...		
}		
},		
...		
}]		
},{	1laneに配備されるFunction1つ分で用意するリソースプールの情報	filter/resize用bitstreamの場合は1laneにつき1Functionのみ

"partitionName": "1",		
...		
},		
]		
}		

フィルタリサイズファンクション名特定用対応情報

※FPGAでフィルタリサイズが高度推論向けか軽量推論向けのどちらなのかを判別するためのマッピング情報。FPGAへの設定パラメータ(出力フレームサイズ)の値でどちらかを判断しているため、設定パラメータ値と対応する推論タイプを記載している。

functionnamemap.json	説明	備考
{		
"sizeList":[{	高度推論用フィルタリサイズ用の情報	高度推論用
"height": 1280,	高度推論用フィルタリサイズの出力フレームの高さのサイズ	固定値
"width": 1280,	高度推論用フィルタリサイズの出力フレームの幅のサイズ	固定値
"functionName":"-high-infer"	高度推論用フィルタリサイズに該当するWBFunctionの名前に付ける文字列	固定値
},{	軽量推論用フィルタリサイズ用の情報	軽量推論用
"height": 416,	軽量推論用フィルタリサイズの出力フレームの高さのサイズ	固定値
"width": 416,	軽量推論用フィルタリサイズの出力フレームの幅のサイズ	固定値
"functionName":"-low-infer"	軽量推論用フィルタリサイズに該当するWBFunctionの名前に付ける文字列	固定値
}]		
}		

タイプ別情報とファイル名のマッピング情報

※自動収集&CM作成機能が内部で使用する、各種タイプ別情報とそのファイル名を結びつける情報。

premadefilelist.json	説明	備考
{		
"region-unique-info" : "region-unique-info.json",	「領域固有情報」とそのファイル名	固定値
"function-unique-info" : "function-unique-info.json",	「Function固有情報-共通属性」とそのファイル名	固定値
"filter-resize-childbs" : "filter-resize-childbs.json",	「Function固有情報-専用属性(filter/resize用)」とそのファイル名	固定値
}		

以下のコンフィグ情報ファイルは環境に合わせた変更が不要でそのまま使える。

GPUFunc用コンフィグ情報

※高度推論用(gpufunc-config-high-infer.json)と軽量推論用(gpufunc-config-low-infer.json)の２種類のコンフィグ情報を用意している。これら以外のGPUFunctionを使う場合は作成が必要。

GPUFuntion用推論処理モジュール(高度推論を実施)のコンフィグ情報

gpufunc-config-high-infer.json	説明	備考（どの値も基本変更不要）
[{	入力側がDMA通信、出力側がRTP通信を行う場合の設定情報	
"rxProtocol": "DMA",	入力側のプロトコル	
"txProtocol":"RTP",	出力側のプロトコル	
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり
"imageURI": "localhost/localhost/gpu_infer_dma:1.0.0",	使用するコンテナのコンテナイメージ名	高度推論を実施するGPUFunc用推論処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"envs":{	使用するコンテナに設定する環境変数群	高度推論を実施するGPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"CUDA_MPS_PIPE_DIRECTORY": "/tmp/nvidia-mps",	MPS機能間での相互通信用のディレクトリのフルパス	
"CUDA_MPS_LOG_DIRECTORY": "/tmp/nvidia-mps",	MPSのログ出力用ディレクトリのフルパス。	
"SHMEM_SECONDARY": "1",	PCIe接続において使用するDPDKで、アプリの起動方法(共有メモリの管理モード)を制御するための情報。	推論アプリ単独で管理するプライマリモードは"0"を、PCIe接続のコントローラと連携管理するセカンダリモードは"1"を設定する。基本"1"を設定する。
"HEIGHT": "1280",	入力映像のフレームサイズ(高さ)。	高度推論なので1280
"WIDTH": "1280"	入力映像のの入力フレームサイズ(幅)。	高度推論なので1280
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "gfunc-hi-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/nvidia/deepstream/deepstream-7.0",	コンテナのワーキングディレクトリ	値は"/opt/nvidia/deepstream/deepstream-X.Y"。"X.Y"の部分は使うdeepstreamのバージョン番号(基本変更不要)
"command": ["sh", "-c"],		実行コマンドの実態は"args"で指定している
"args":["cd /opt/DeepStream-Yolo && gst-launch-1.0 -ev fpgasrc !", "'video/x-raw,format=(string)BGR,%WIDTH%,%HEIGHT%'", "! nvvideoconvert ! 'video/x-raw(memory:NVMM), format=(string)RGBA'", "! m.sink_0 nvstreammux name=m nvbuf-memory-type=0 batch-size=1", "%WIDTH%", "%HEIGHT%", "! queue ! nvinfer config-file-path=./config_infer_primary_yoloV4_p6_th020_040.txt batch-size=1", "model-engine-file=./model_b1_gpu0_fp16.engine ! queue ! nvdsosd process-mode=1 ! nvvideoconvert !", "'video/x-raw, format=(string)BGR' ! videoconvert ! queue ! perf ! rtpvrawpay ! udpsink", "%OUTPUTIP%", "%OUTPUTPORT%", "sync=true"],	コマンド実行時に渡す引数	コンテナで実行する高度処理モジュールを軽量推論で実施するためのGstremerプラグインの実行コマンドと引数
"securityContext":{		
"privileged": true		値は"true"固定
},		
"volumeMounts":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolumeMount	起動するコンテナは1台なので固定で良い
"mountPath": "/dev/hugepages"		値は"/dev/hugepages"固定
},{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolumeMount	
"mountPath": "/tmp/nvidia-mps"		環境変数"CUDA_MPS_PIPE_DIRECTORY"と同じ値
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolumeMount	
"mountPath": "/var/run/dpdk"		値は"var-run-dpdk"固定
}],		
"resources":{		
"requests":{		
"memory": "32Gi"		共有メモリ(hugepage)用。k8sの仕様「hugepageを使用する際には、CPUやメモリ(最低どちらか１つ)をリクエストしなければいけない」に対応するため設定。 値は任意で良いので"32Gi"固定で良い
},		
"limits":{		
"hugepages-1Gi": "1Gi"		値は"1Gi"固定
}		
}		
}],		
"volumes":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolume	上記volumeMounts."hugepage-1gi"と同じ値
"hostPath":		
{"path": "/dev/hugepages"}		上記volumeMounts."hugepage-1gi"の"mountPath"と同じ値
},{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolume	上記volumeMounts."host-nvidia-mps"と同じ値
"hostPath":		
{"path": "/tmp/nvidia-mps"}		上記volumeMounts."host-nvidia-mps"の"mountPath"と同じ値
},{		

"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolume	上記volumeMounts."dpdk"と同じ値
"hostPath":		
{"path": "/var/run/dpdk"}		上記volumeMounts."dpdk"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		
"restartPolicy": "Always"		
}		
}		
},		
{	入力側がTCP通信、出力側がRTP通信を行う場合の設定情報	
"rxProtocol": "TCP",	入力側のプロトコル	
"txProtocol":"RTP",	出力側のプロトコル	
"imageURI": "localhost/gpu_infer_tcp:1.0.0",	使用するコンテナのコンテナイメージ名	高度推論を実施するGPUFunc用推論処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"envs":{	使用するコンテナに設定する環境変数群	高度推論を実施するGPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"CUDA_MPS_PIPE_DIRECTORY": "/tmp/nvidia-mps",	MPS機能間での相互通信用のディレクトリのフルパス	
"CUDA_MPS_LOG_DIRECTORY": "/tmp/nvidia-mps",	MPSのログ出力用ディレクトリのフルパス。	
"GST_PLUGIN_PATH": "/opt/nvidia/deepstream/deepstream-7.0/sample-functions/functions/gpu_infer_tcp_plugins/fpga_depayloader",	コンテナ内でのGstreamerプラグインの格納ディレクトリ。	
"HEIGHT": "1280",	入力映像のフレームサイズ(高さ)。	高度推論なので1280
"WIDTH": "1280"	入力映像のの入力フレームサイズ(幅)。	高度推論なので1280
},		
"template":{		
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "gfunc-hi-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/nvidia/deepstream/deepstream-7.0",	コンテナのワーキングディレクトリ	値は"/opt/nvidia/deepstream/deepstream-X.Y"。"X.Y"の部分は使うdeepstreamのバージョン番号(基本変更不要)
"command": ["sh", "-c"],		実行コマンドの実態は"args"で指定している
"args":["cd /opt/DeepStream-Yolo && gst-launch-1.0 -ev fpgadepay", "%INPUTIP%", "%INPUTPORT%", "! 'video/x-raw,format=(string)BGR,%WIDTH%,%HEIGHT%'", "! nvvideoconvert ! 'video/x-raw(memory:NVMM), format=(string)RGBA'", "! m.sink_0 nvstreammux name=m nvbuf-memory-type=0 batch-size=1", "%WIDTH%", "%HEIGHT%", "! queue ! nvinfer config-file-path=./config_infer_primary_yoloV4_p6_th020_040.txt batch-size=1", "model-engine-file=./model_b1_gpu0_fp16.engine ! queue ! nvdsosd process-mode=1 ! nvvideoconvert !", "'video/x-raw, format=(string)BGR' ! videoconvert ! queue ! perf ! rtpvrawpay ! udpsink", "%OUTPUTIP%", "%OUTPUTPORT%", "sync=true"].	コマンド実行時に渡す引数	コンテナで実行する高度処理モジュールを軽量推論で実施するためのGstremerプラグインの実行コマンドと引数
"securityContext":{		
"privileged": true		
},		
"volumeMounts":[{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolumeMount	
"mountPath": "/tmp/nvidia-mps"		環境変数"CUDA_MPS_PIPE_DIRECTORY"と同じ値
}]		
}],		
"volumes":[{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolume	上記volumeMounts."host-nvidia-mps"と同じ
"hostPath":		
{"path": "/tmp/nvidia-mps"}		上記volumeMounts."host-nvidia-mps"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		
"restartPolicy": "Always"		
}		
}		
}]		

GPUFuntion用推論処理モジュール(軽量推論を実施)のコンフィグ情報

gpufunc-config-low-infer.json	説明	備考（どの値も基本変更不要）
[{	入力側がDMA通信、出力側がRTP通信を行う場合の設定情報	
"rxProtocol": "DMA",	入力側のプロトコル	
"txProtocol":"RTP",	出力側のプロトコル	
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり
"imageURI": "localhost/gpu_infer_dma:1.0.0",	使用するコンテナのコンテナイメージ名	軽量推論を実施するGPUFunc用推論処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定

"envs":{	使用するコンテナに設定する環境変数群	軽量推論を実施するGPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"CUDA_MPS_PIPE_DIRECTORY": "/tmp/nvidia-mps",	MPS機能間での相互通信用のディレクトリのフルパス	
"CUDA_MPS_LOG_DIRECTORY": "/tmp/nvidia-mps",	MPSのログ出力用ディレクトリのフルパス。	
"SHMEM_SECONDARY": "1",	PCIe接続において使用するDPDKで、アプリの起動方法(共有メモリの管理モード)を制	推論アプリ単独で管理するプライマリモードは"0"を、PCIe接続のコントローラと連携管理するセカンダリモードは"1"を設定する。基本"1"を設定する。
"HEIGHT": "416",	入力映像のフレームサイズ(高さ)。	軽量推論なので416
"WIDTH": "416"	入力映像のの入力フレームサイズ(幅)。	軽量推論なので416
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "gfunc-n02-lo-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/nvidia/deepstream/deepstream-7.0",	コンテナのワーキングディレクトリ	値は"/opt/nvidia/deepstream/deepstream-X.Y"。"X.Y"の部分は使うdeepstreamのバージョン番号(基本変更不要)
"command": ["sh", "-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["cd /opt/nvidia/deepstream/deepstream-7.0/sources/objectDetector_Yolo/ && gst-launch-1.0 -ev fpgasrc !", "'video/x-raw,format=(string)BGR,%WIDTH%,%HEIGHT%'", "! nvvideoconvert ! 'video/x-raw(memory:NVMM), format=(string)RGBA'", "! m.sink_0 nvstreammux name=m nvbuf-memory-type=0 batch-size=1", "%WIDTH%", "%HEIGHT%", "! queue ! nvinfer config-file-path=./config_infer_primary_yoloV3_tiny.txt", "batch-size=1 model-engine-file=./model_b1_gpu0_int8.engine ! queue ! nvvideoconvert !", "'video/x-raw, format=(string)BGR' ! videoconvert ! queue ! perf ! rtpvrawpay ! udpsink", "%OUTPUTIP%", "%OUTPUTPORT%", "sync=true"],	コマンド実行時に渡す引数	コンテナで実行する推論処理モジュールを軽量推論で実施するためのGstremerプラグインの実行コマンドと引数
"securityContext":{		
"privileged": true		値は"true"固定
},		
"volumeMounts":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolumeMount	値は"hugepage-1gi"固定
"mountPath": "/dev/hugepages"		値は"/dev/hugepages"固定
},{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolumeMount	
"mountPath": "/tmp/nvidia-mps"		環境変数"CUDA_MPS_PIPE_DIRECTORY"と同じ値
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolumeMount	
"mountPath": "/var/run/dpdk"		値は"var-run-dpdk"固定
}],		
"resources":{		
"requests":{		
"memory": "32Gi"		共有メモリ(hugepage)用。k8sの仕様「hugepageを使用する際には、CPUやメモリ(最低どちらか１つ)をリクエストしなければいけない」に対応するため設定。 値は任意で良いので"32Gi"固定で良い
},		
"limits":{		
"hugepages-1Gi": "1Gi"	hugepageの1枚分のページサイズ	値は"1Gi"固定
}		
}		
}],		
"volumes":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolume	上記volumeMounts."hugepage-1gi"と同じ値
"hostPath":		
{"path": "/dev/hugepages"}		上記volumeMounts."hugepage-1gi"の"mountPath"と同じ値
},{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolume	上記volumeMounts."host-nvidia-mps"と同じ値
"hostPath":		
{"path": "/tmp/nvidia-mps"}		上記volumeMounts."host-nvidia-mps"の"mountPath"と同じ値
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolume	上記volumeMounts."dpdk"と同じ値
"hostPath":		
{"path": "/var/run/dpdk"}		上記volumeMounts."dpdk"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
},		
{	入力側がTCP通信、出力側がRTP通信を行う場合の設定情報	
"rxProtocol": "TCP",	入力側のプロトコル	
"txProtocol":"RTP",	出力側のプロトコル	
"imageURI": "localhost/gpu_infer_tcp:1.0.0",	使用するコンテナのコンテナイメージ名	軽量推論を実施するGPUFunc用推論処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は“sriov”のみを前提としているため、設定値は"sriov"固定

"envs":{	使用するコンテナに設定する環境変数群	軽量推論を実施するGPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"CUDA_MPS_PIPE_DIRECTORY": "/tmp/nvidia-mps",	MPS機能間での相互通信用のディレクトリのフルパス	
"CUDA_MPS_LOG_DIRECTORY": "/tmp/nvidia-mps",	MPSのログ出力用ディレクトリのフルパス。	
"GST_PLUGIN_PATH": "/opt/nvidia/deepstream/deepstream-7.0/sample-functions/functions/gpu_infer_tcp_plugins/fpga_depayloader",	コンテナ内でのGstreamerプラグインの格納ディレクトリ。	
"HEIGHT": "416",	入力映像のフレームサイズ(高さ)。	軽量推論なので416
"WIDTH": "416"	入力映像のの入力フレームサイズ(幅)。	軽量推論なので416
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "gfunc-n02-lo-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/nvidia/deepstream/deepstream-7.0",	コンテナのワーキングディレクトリ	値は"/opt/nvidia/deepstream/deepstream-X.Y"。"X.Y"の部分は使うdeepstreamのバージョン番号(基本変更不要)
"command": ["sh", "-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["cd /opt/nvidia/deepstream/deepstream-7.0/sources/objectDetector_Yolo/ && gst-launch-1.0 -ev fpgadepay", "%INPUTIP%", "%INPUTPORT%", "! 'video/x-raw,format=(string)BGR,%WIDTH%,%HEIGHT%'", "! nvvideoconvert ! 'video/x-raw(memory:NVMM), format=(string)RGBA'", "! m.sink_0 nvstreammux name=m nvbuf-memory-type=0 batch-size=1", "%WIDTH%", "%HEIGHT%", "! queue ! nvinfer config-file-path=./config_infer_primary_yoloV3_tiny.txt", "batch-size=1 model-engine-file=./model_b1_gpu0_int8.engine ! queue ! nvvideoconvert !", "'video/x-raw, format=(string)BGR' ! videoconvert ! queue ! perf ! rtpvrawpay ! udpsink", "%OUTPUTIP%", "%OUTPUTPORT%", "sync=true"],	コマンド実行時に渡す引数	コンテナで実行する推論処理モジュールを軽量推論で実施するためのGstremerプラグインの実行コマンドと引数
"securityContext":{		
"privileged": true		
},		
"volumeMounts":[{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolumeMount	
"mountPath": "/tmp/nvidia-mps"		環境変数"CUDA_MPS_PIPE_DIRECTORY"と同じ値
}]		
}],		
"volumes":[{		
"name": "host-nvidia-mps",	MPS用。MPS機能間で相互通信するためのディレクトリ用のVolume	上記volumeMounts."host-nvidia-mps"と同じ
"hostPath":		
{"path": "/tmp/nvidia-mps"}		上記volumeMounts."host-nvidia-mps"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

FPGAFunc用コンフィグ情報

※高度推論向けのフィルタリサイズ用(fpgafunc-config-filter-resize-high-infer.json)と軽量推論向けのフィルタリサイズ用(fpgafunc-config-filter-resize-low-infer.json)の２種類のコンフィグ情報を用意している。これら以外のFPGAFunctionを使う場合は作成が必要。
(フィルタリサイズ用が高度推論向けと軽量推論向けに分かれている様に)入力パラメータの値ごとに別途コンフィグ情報を作成する。（同じ処理でも入力パラメータの値が異なると別のFPGAFunctionとして定義する想定なため）

FPGAFunc用フィルタリサイズ処理モジュール(高度推論向けの処理を実施)のコンフィグ情報

fpgafunc-config-filter-resize-high-infer.json	説明	備考（どの値も基本変更不要）
{		
"parentBitstream": {	使用する親Bitstreamの情報	
"file": "OpenKasugai-fpga-example-design-1.0.0-1.mcs",	親Bitstreamのファイル名	
"id": "0100001c"	親BitstreamのビットストリームID	
},		
"childBitstream": {	使用する子Bitstreamの情報	
"file": "OpenKasugai-fpga-example-design-1.0.0-2.bit",	子Bitstreamのファイル名	
"id": "0100001c"	子BitstreamのビットストリームID	
},		
"parameters": {	使用するFPGAの子bslに設定する環境変数群	高度推論用フィルタリサイズ処理モジュール用bitstreamに設定するパラメータ群
"functions": {	設定先のモジュール名	フィルタリサイズの場合はfunctionsモジュールにのみ処理モジュール用パラメータを設定すれば良い
"i_width": 3840,	入力フレームの幅のサイズ	高度用でも軽量用でも3840
"i_height": 2160,	入力フレームの高さのサイズ	高度用でも軽量用でも2140
"o_width": 1280,	出力フレームの幅のサイズ	高度推論用なので1280
"o_height": 1280	出力フレームの高さのサイズ	高度推論用なので1280
}		
},		
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり。(基本"256"で良い)

"functionDedicatedInfo": "filter-resize-ch"		
}		

FPGAFunc用フィルタリサイズ処理モジュール(軽量推論向けの処理を実施)のコンフィグ情報

fpgafunc-config-filter-resize-low-infer.json	説明	備考（どの値も基本変更不要）
{		
"parentBitstream": {	使用する親Bitstreamの情報	
"file": "OpenKasugai-fpga-example-design-1.0.0-1.mcs",	親Bitstreamのファイル名	
"id": "0100001c"	親BitstreamのビットストリームID	
},		
"childBitstream": {	使用する子Bitstreamの情報	
"file": "OpenKasugai-fpga-example-design-1.0.0-2.bit",	子Bitstreamのファイル名	
"id": "0100001c"	子BitstreamのビットストリームID	
},		
"parameters": {	使用するFPGAの子bsに設定する環境変数群	軽量推論用フィルタリサイズ処理モジュール用bitstreamに設定するパラメータ群
"functions": {	設定先のモジュール名	フィルタリサイズの場合はfunctionsモジュールにのみ処理モジュール用パラメータを設定すれば良い
"i_width": 3840,	入力フレームの幅のサイズ	高度用でも軽量用でも3840
"i_height": 2140,	入力フレームの高さのサイズ	高度用でも軽量用でも2140
"o_width": 416,	出力フレームの幅のサイズ	軽量推論用なので416
"o_height": 416	出力フレームの高さのサイズ	軽量推論用なので416
}		
},		
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり。(基本"256"で良い)
"functionDedicatedInfo": "filter-resize-ch"		
}		

CPUFunc用コンフィグ情報

※デコード用(cpufunc-config-decode.json)と高度推論向けのフィルタリサイズ用(cpufunc-config-filter-resize-high-infer.json)と軽量推論向けのフィルタリサイズ用(cpufunc-config-filter-resize-low-infer.json)とコピー分岐用(cpufunc-config-copy-branch.json)とGlue用(cpufunc-config-glue-fdma-to-tcp.json)の5種類のコンフィグ情報を用意している。これら以外のCPUFunctionを使う場合は作成が必要。
なお、入力パラメータの値ごとに別途コンフィグ情報を作成する。（入力パラメータの値が異なると別のCPUFunctionとして定義する想定なため）

CPUFuntion用デコード処理モジュールのコンフィグ情報

cpufunc-config-decode.json	説明	備考（どの値も基本変更不要）
[{	入力側がRTP通信、出力側がDMA通信を行う場合の設定情報	
"rxProtocol":"RTP",	入力側のプロトコル	"RTP"固定
"txProtocol":"DMA",	出力側のプロトコル	"DMA"固定
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり。(基本"256"で良い)
"imageURI": "localhost/cpu_decode:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用デコード処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"envs":{	使用するコンテナに設定する環境変数群	CPUFunc用デコード処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"DECENV_APPLOG_LEVEL": "6",	ログレベル	値は特に変更不要
"DECENV_FRAME_WIDTH": "3840",	入力映像のフレームサイズ(幅)	サンプルユースケースでは3840固定
"DECENV_FRAME_HEIGHT": "2160",	入力映像のフレームサイズ(高さ)	サンプルユースケースでは2160固定
"DECENV_VIDEO_CONNECT_LIMIT": "0",	映像送信元との連続接続回数。	デフォルト値"0"のままで良い
"DECENV_VIDEOSRC_PROTOCOL": "RTP",	受信するプロトコル	"RTP"固定
"DECENV_OUTDST_PROTOCOL": "DMA"	送信するプロトコル	"DMA"固定
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "cfunc-1",		起動するコンテナは1台なので固定で良い
"command": ["sh","-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["./sample-functions/functions/cpu_decode/build/cpu_decode-shared"],	コマンド実行時に渡す引数	CPUFunc用デコード処理モジュールをコンテナで実行するためのコマンド
"securityContext":{		
"privileged": true		値は"true"固定
},		
"volumeMounts":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolumeMount	値は"hugepage-1gi"固定
"mountPath": "/dev/hugepages"		値は"/dev/hugepages"固定
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolumeMount	
"mountPath": "/var/run/dpdk"		値は"var-run-dpdk"固定
}],		
"resources":{		
"requests":{		
"memory": "32Gi"		共有メモリ(hugepage)用。k8sの仕様「hugepageを使用する際には、CPUやメモリ(最低どちらか１つ)をリクエストしなければいけない」に対応するため設定。 値は任意で良いので"32Gi"固定で良い
},		
"limits":{		

"hugepages-1Gi": "1Gi"	hugepageの1枚分のページサイズ	値は"1Gi"固定
}		
}		
}],		
"volumes":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolume	上記volumeMounts."hugepage-1gi"と同じ値
"hostPath":		
{"path": "/dev/hugepages"}		上記volumeMounts."hugepage-1gi"の"mountPath"と同じ値
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolume	上記volumeMounts."dpdk"と同じ値
"hostPath":		
{"path": "/var/run/dpdk"}		上記volumeMounts."dpdk"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
},		
{	入力側がRTP通信、出力側がTCP通信を行う場合の設定情報	
"rxProtocol":"RTP",	受信側(RTP接続)のプロトコル	"RTP"固定
"txProtocol":"TCP",	送信側(Eth接続)のプロトコル	"TCP"固定
"imageURI": "localhost/cpu_decode:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用デコード処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"envs":{	使用するコンテナに設定する環境変数群	CPUFunc用デコード処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"DECENV_APPLOG_LEVEL": "6",	ログレベル	値は特に変更不要
"DECENV_FRAME_WIDTH": "3840",	入力映像のフレームサイズ(幅)	サンプルユースケースでは3840固定
"DECENV_FRAME_HEIGHT": "2160",	入力映像のフレームサイズ(高さ)	サンプルユースケースでは2160固定
"DECENV_VIDEO_CONNECT_LIMIT": "0",	映像送信元との連続接続回数。	デフォルト値"0"のままで良い
"DECENV_VIDEOSRC_PROTOCOL": "RTP",	受信するプロトコル	"RTP"固定
"DECENV_OUTDST_PROTOCOL": "TCP"	送信するプロトコル	"TCP"固定
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "cfunc-1",		起動するコンテナは1台なので固定で良い
"command": ["sh","-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["./sample-functions/functions/cpu_decode/build/cpu_decode-shared"],	コマンド実行時に渡す引数	内容はCPUFunc用デコードの実行コマンド(実行ファイルのパス)
"securityContext":{		
"privileged": true		値は"true"固定
}		
}],		
"hostNetwork": false,	Dockerホスト側のネットワークスタックを使用するためのコンテナのネットワーク設定	現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

フィルタリサイズ処理モジュール(高度推論向けの処理を実施)向けのCPUFuntion用コンフィグ情報

cpufunc-config-filter-resize-high-infer.json	説明	備考（どの値も基本変更不要）
[{	入出力ともTCP通信を行う場合の設定情報	現状は入出力ともTCP通信向けにのみ対応しているので、この場合の設定情報しかない
"rxProtocol":"TCP",	入力側のプロトコル	"TCP"固定
"txProtocol":"TCP",	出力側のプロトコル	"TCP"固定
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"imageURI": "localhost/cpu_filter_resize:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用デコード処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"envs":{	使用するコンテナに設定する環境変数群	高度推論を実施するCPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"FRENV_APPLOG_LEVEL": "DEBUG",	ログレベル	値は特に変更不要
"FRENV_INPUT_WIDTH": "3840",	入力映像のフレームサイズ(幅)	サンプルユースケースでは3840固定
"FRENV_INPUT_HEIGHT": "2160",	入力映像のフレームサイズ(高さ)	サンプルユースケースでは2160固定
"FRENV_OUTPUT_WIDTH": "1280",	出力映像のフレームサイズ(幅)	高度推論なので1280
"FRENV_OUTPUT_HEIGHT": "1280"	出力映像のフレームサイズ(高さ)	高度推論なので1280
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "fr",		起動するコンテナは1台なので固定で良い

"command": ["python", "fr.py", "--in_port=$(FRENV_INPUT_PORT)", "--out_addr=$(FRENV_OUTPUT_IP)", "--out_port=$(FRENV_OUTPUT_PORT)", "--in_width=$(FRENV_INPUT_WIDTH)", "--in_height=$(FRENV_INPUT_HEIGHT)", "--out_width=$(FRENV_OUTPUT_WIDTH)", "--out_height=$(FRENV_OUTPUT_HEIGHT)", "--loglevel=$(FRENV_APPLOG_LEVEL)"],	使用するコンテナでの実行コマンド	CPUFunc用フィルタリサイズ処理モジュールを高度推論向けに実施するためにコンテナで実行するためのコマンド
"securityContext":{		
"privileged": true		値は"true"固定
}		
}],		
"hostNetwork": false,	Dockerホスト側のネットワークスタックを使用するためのコンテナのネットワーク設定	現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

フィルタリサイズ処理モジュール(軽量推論向けの処理を実施)向けのCPUFuntion用コンフィグ情報

cpufunc-config-filter-resize-low-infer.json	説明	備考（どの値も基本変更不要）
[{	入出力ともTCP通信を行う場合の設定情報	現状は入出力ともTCP通信向けにのみ対応しているので、この場合の設定情報しかない
"rxProtocol":"TCP",	入力側のプロトコル	"TCP"固定
"txProtocol":"TCP",	出力側のプロトコル	"TCP"固定
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"imageURI": "localhost/cpu_filter_resize:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用デコード処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"envs":{	使用するコンテナに設定する環境変数群	高度推論を実施するCPUFunc用推論処理モジュールのコンテナ実行に必要な環境変数で、pod用テンプレート配下のspec.containers[i].envに設定される
"FRENV_APPLOG_LEVEL": "DEBUG",	ログレベル	値は特に変更不要
"FRENV_INPUT_WIDTH": "3840",	入力映像のフレームサイズ(幅)	サンプルユースケースでは3840固定
"FRENV_INPUT_HEIGHT": "2160",	入力映像のフレームサイズ(高さ)	サンプルユースケースでは2160固定
"FRENV_OUTPUT_WIDTH": "416",	出力映像のフレームサイズ(幅)	高度推論なので1280
"FRENV_OUTPUT_HEIGHT": "416"	出力映像のフレームサイズ(高さ)	高度推論なので1280
},		
"template":{	作成するPodのテンプレートデータ	コンテナに設定する環境変数(env)はテンプレートの外部(上述のenvs)に記載している
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "fr",		起動するコンテナは1台なので固定で良い
"command": ["python", "fr.py", "--in_port=$(FRENV_INPUT_PORT)", "--out_addr=$(FRENV_OUTPUT_IP)", "--out_port=$(FRENV_OUTPUT_PORT)", "--in_width=$(FRENV_INPUT_WIDTH)", "--in_height=$(FRENV_INPUT_HEIGHT)", "--out_width=$(FRENV_OUTPUT_WIDTH)", "--out_height=$(FRENV_OUTPUT_HEIGHT)", "--loglevel=$(FRENV_APPLOG_LEVEL)"],	使用するコンテナでの実行コマンド	CPUFunc用フィルタリサイズ処理モジュールを軽量推論向けに実施するためにコンテナで実行するためのコマンド
"securityContext":{		
"privileged": true		値は"true"固定
}		
}],		
"hostNetwork": false,	Dockerホスト側のネットワークスタックを使用するためのコンテナのネットワーク設定	現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

コピー分岐処理モジュール向けのCPUFuntion用コンフィグ情報

cpufunc-config-copy-branch.json	説明	備考（どの値も基本変更不要）
[{	入出力ともTCP通信を行う場合の設定情報	現状は入出力ともTCP通信向けにのみ対応しているので、この場合の設定情報しかない
"rxProtocol":"TCP",	入力側のプロトコル	"TCP"固定
"txProtocol":"TCP",	出力側のプロトコル	"TCP"固定
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"copyMemorySize": "1024",	TCP受信データ格納メモリ1領域あたりのメモリサイズ情報(Byte)	デフォルト値"1024"固定
"imageURI": "localhost/cpu_copy_branch:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用コピー分岐処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"template":{	作成するPodのテンプレートデータ	
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		

"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "cfunc-copy-branch-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/openkasugai-controller/sample-functions/functions-	コンテナのワーキングディレクトリ	値は"/opt/openkasugai-controller/sample-functions/functions-ext/cpu_copy_branch"固定
"command": ["sh","-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["./copy_branch", "%RECEIVING%", "%NUM%", "%FORWARDING%", "%MEMSIZE%"],	コマンド実行時に渡す引数	CPUFunc用コピー分岐処理モジュールの実行コマンドとその引数
"securityContext":{		
"privileged": true		値は"true"固定
}		
}],		
"hostNetwork": false,	Dockerホスト側のネットワークスタックを使用するためのコンテナのネットワーク設定	現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

glue(DMA→TCP変換)処理モジュール向けのCPUFuntion用コンフィグ情報

cpufunc-config-glue-fdma-to-tcp.json	**説明**	**備考（どの値も基本変更不要）**
[{	入力側がDMA通信、出力側がTCP通信を行う場合の設定情報	DMA→TCPの変換用の処理モジュールなのでこの場合の設定情報しかない
"rxProtocol":"DMA",	入力側のプロトコル	"DMA"固定
"txProtocol":"TCP",	出力側のプロトコル	"TCP"固定
"sharedMemoryMiB": 256,	PCIe接続時に使用するHugePage内確保サイズ[MegaByte]	確保するサイズは2のべき乗である必要あり。(基本"256"で良い)
"imageURI": "localhost/cpu_glue_dma_tcp:1.0.0",	使用するコンテナのコンテナイメージ名	CPUFunc用glue(DMA→TCP変換)処理モジュールのコンテナイメージ名。タグの数字はバージョン番号(基本変更不要)
"additionalNetwork": true,	ファンクションの2nd NICの利用の有無	現状は2nd NIC利用を前提としているため値は"true"固定
"virtualNetworkDeviceDriverType": "sriov",	ファンクションが2nd NICで利用する仮想NWデバイス用ドライバ	現状は"sriov"のみを前提としているため、設定値は"sriov"固定
"template":{	作成するPodのテンプレートデータ	
"apiVersion": "v1",		
"kind": "Pod",		
"spec":{		
"containers":[{	Podで起動するコンテナに関する設定情報	
"name": "cfunc-glue-fdma-to-tcp-1",		起動するコンテナは1台なので固定で良い
"workingDir": "/opt/openkasugai-controller/sample-functions/functions-	コンテナのワーキングディレクトリ	値は"/opt/openkasugai-controller/sample-functions/functions-ext/cpu_glue_dma_tcp"固定
"command": ["sh","-c"],	使用するコンテナでの実行コマンド	実行コマンドの実態は"args"で指定している
"args":["./build/glue", "%FORWARDING%", "%WIDTH%", "%HEIGHT%"],	コマンド実行時に渡す引数	CPUFunc用glue(DMA→TCP変換)処理モジュールの実行コマンドとその引数
"securityContext":{		
"privileged": true		値は"true"固定
},		
"volumeMounts":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolumeMount	値は"hugepage-1gi"固定
"mountPath": "/dev/hugepages"		値は"/dev/hugepages"固定
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolumeMount	
"mountPath": "/var/run/dpdk"		値は"var-run-dpdk"固定
}],		
"resources":{		
"requests":{		
"memory": "32Gi"		共有メモリ(hugepage)用。k8sの仕様「hugepageを使用する際には、CPUやメモリ(最低どちらか１つ)をリクエストしなければいけない」に対応するため設定。 値は任意で良いので"32Gi"固定で良い
},		
"limits":{		
"hugepages-1Gi": "1Gi"	hugepageの1枚分のページサイズ	値は"1Gi"固定
}		
}		
}],		
"volumes":[{		
"name": "hugepage-1gi",	入力側のPCIe接続で使用する/dev/hugepagesディレクトリ用のVolume	上記volumeMounts."hugepage-1gi"と同じ値
"hostPath":		
{"path": "/dev/hugepages"}		上記volumeMounts."hugepage-1gi"の"mountPath"と同じ値
},{		
"name": "dpdk",	入力側のPCIe接続で使用するDPDKで使用するディレクトリ用のVolume	上記volumeMounts."dpdk"と同じ値
"hostPath":		
{"path": "/var/run/dpdk"}		上記volumeMounts."dpdk"の"mountPath"と同じ値
}],		
"hostNetwork": false,		現状は2nd NIC利用を前提としているため値は"false"で良い。
"hostIPC": true,		値は"true"固定
"restartPolicy": "Always"		値は"Always"固定
}		
}		
}]		

DeviceInfoコントローラ用（赤太字箇所が環境に合わせて変更するポイント）

crc_deviceinfo_daemonset.yaml	説明	備考
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-deviceinfo-daemon		
spec:		
selector:		
matchLabels:		
app: crc-deviceinfo-daemon		
template:		
metadata:		
labels:		
app: crc-deviceinfo-daemon		
spec:		
containers:		
- name: deviceinfo-container0		
image: localhost/deviceinfo:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_CLUSTERNAME	当該環境のk8sクラスタ名に関する環境変数	固定値
value: default		環境のクラスタ名に合わせて変更
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-deviceinfo-daemon		
volumes:		
- name: crc-deviceinfo-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		

PCIeConnectionコントローラ用（赤太字箇所が環境に合わせて変更するポイント）

crc_pcieconnection_daemonset.yaml	説明	備考
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-pcieconnection-daemon		
spec:		
selector:		
matchLabels:		
app: crc-pcieconnection-daemon		
template:		
metadata:		
labels:		
app: crc-pcieconnection-daemon		
spec:		
containers:		
- name: pcieconnection-container0		
image: localhost/pcieconnection:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-pcieconnection-daemon		
- mountPath: /var/run/dpdk	PCIe接続の際に使用するDPDKで使用するディレクトリを設定	DPDKで使用するディレクトリを設定
name: var-run-dpdk		
- name: hugepage-1gi	hugepageとして用意したディレクトリの設定	想定環境では、HugePageとして使用するPCIe接続するので、その際の共有メモリ1枚分のページサイズを1GiBとして使っている。 当該workerノードにて設定している共有メモリ1枚分のページサイズによって、数字部分を変更する。
mountPath: /dev/hugepages		hugepageとして用意したディレクトリのファイルパスを設定
resources:		
limits:		
hugepages-1Gi: 16Gi		osで設定したhugepageのサイズ。基本は左記の値のままで良い。
requests:		
hugepages-1Gi: 16Gi		osで設定したhugepageのサイズ。基本は左記の値のままで良い。
memory: 1Gi		
volumes:		

- name: crc-pcieconnection-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		
- name: var-run-dpdk	PCIe接続の際に使用するDPDKで使用するディレクトリを設定	
hostPath:		
path: /var/run/dpdk		DPDKで使用するディレクトリを設定
type: DirectoryOrCreate		
- name: hugepage-1gi	hugepageとして用意したディレクトリの設定	想定環境では、HugePageとして使用するPCIe接続するので、その際の共有メモリ1枚分のページサイズを1GiBとして使っている。 当該workerノードにて設定している共有メモリ1枚分のページサイズによって、数字部分を変更する。
hostPath:		
path: /dev/hugepages		hugepageとして用意したディレクトリのファイルパスを設定
type: DirectoryOrCreate		

EthernetConnectionコントローラ用（特に環境に合わせて変更する箇所は無い）

crc_ethernetconnection_daemonset.yaml	**説明**	**備考**
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-ethernetconnection-daemon		
spec:		
selector:		
matchLabels:		
app: crc-ethernetconnection-daemon		
template:		
metadata:		
labels:		
app: crc-ethernetconnection-daemon		
spec:		
containers:		
- name: ethernetconnection-container0		
image: localhost/ethernetconnection:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-ethernetconnection-daemon		
volumes:		
- name: crc-ethernetconnection-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		

FPGAFunctionコントローラ用（特に環境に合わせて変更する箇所は無い）

crc_fpgafunction_daemonset.yaml	**説明**	**備考**
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-fpgafunction-daemon		
spec:		
selector:		
matchLabels:		
app: crc-fpgafunction-daemon		
template:		
metadata:		
labels:		
app: crc-fpgafunction-daemon		
spec:		
containers:		
- name: fpgafunction-container0		
image: localhost/fpgafunction:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-fpgafunction-daemon		
volumes:		

- name: crc-fpgafunction-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		

GPUFunctionコントローラ用（特に環境に合わせて変更する箇所は無い）

crc_gpufunction_daemonset.yaml	説明	備考
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-gpufunction-daemon		
spec:		
selector:		
matchLabels:		
app: crc-gpufunction-daemon		
template:		
metadata:		
labels:		
app: crc-gpufunction-daemon		
spec:		
containers:		
- name: gpufunction-container0		
image: localhost/gpufunction:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-gpufunc-daemon		
volumes:		
- name: crc-gpufunc-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		

CPUFunctionコントローラ用（特に環境に合わせて変更する箇所は無い）

crc_cpufunction_daemonset.yaml	説明	備考
apiVersion: apps/v1		
kind: DaemonSet		
metadata:		
name: crc-cpufunction-daemon		
spec:		
selector:		
matchLabels:		
app: crc-cpufunction-daemon		
template:		
metadata:		
labels:		
app: crc-cpufunction-daemon		
spec:		
containers:		
- name: cpufunction-container0		
image: localhost/cpufunction:1.0.0		
imagePullPolicy: IfNotPresent		
securityContext:		
privileged: true		
args:		
- "--kubeconfig=/kube/config"		
env:		
- name: K8S_NODENAME		
valueFrom:		
fieldRef:		
fieldPath: spec.nodeName		
volumeMounts:		
- mountPath: /kube/config		
name: crc-cpufunc-daemon		
volumes:		
- name: crc-cpufunc-daemon		
hostPath:		
path: /etc/k8s_worker/config		
type: File		