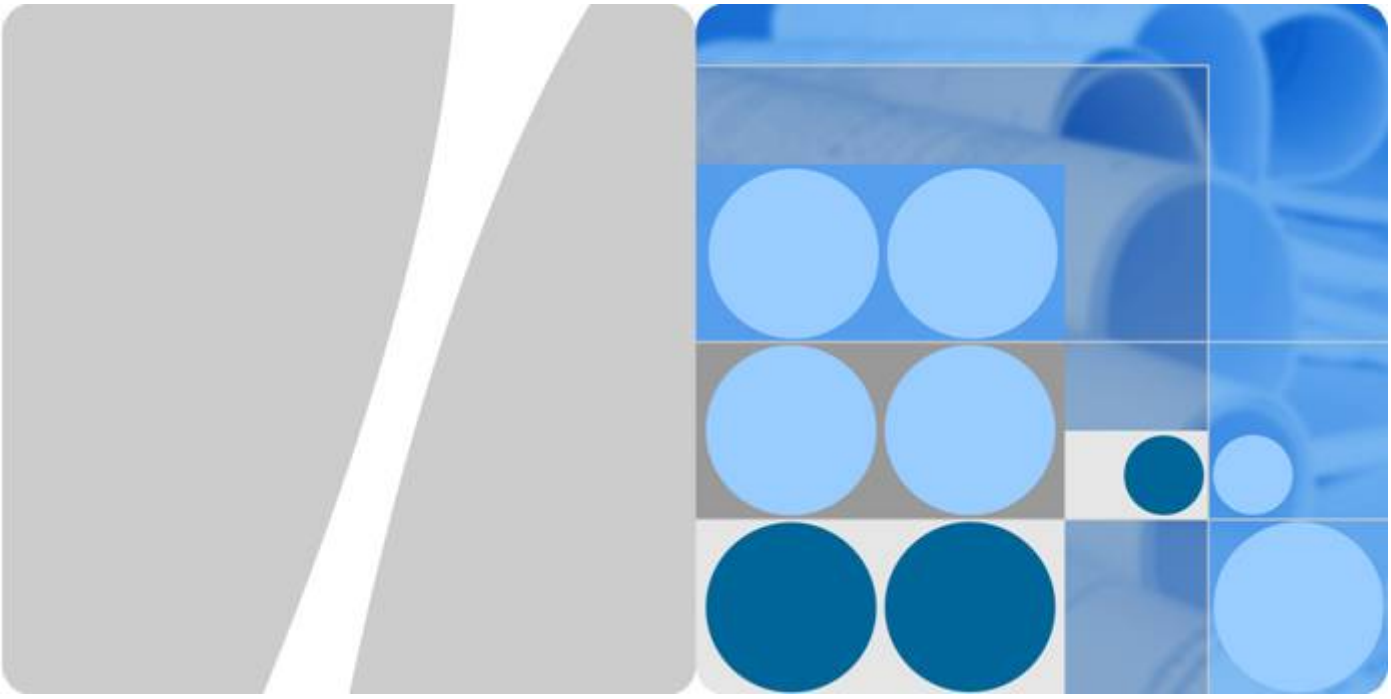

Ascend 310

V100R001

Framework API 参考

文档版本　01

发布日期　2019-03-12

华为技术有限公司

HUAWEI

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为技术有限公司

地址： 深圳市龙岗区坂田华为总部办公楼 邮编：518129

网址： http://www.huawei.com

客户服务邮箱： support@huawei.com

客户服务电话： 4008302118

目 录

1 简介

本文介绍编译自定义算子时所依赖的Framework组件的接口，接口主要包括非类成员函数、类成员函数，同时本文还介绍了与接口相关的宏以及离线模型的proto定义。

2 非类成员函数

2.1 custom_op_run 函数

函数功能

自定义算子的单算子调测入口函数。

函数原型

ErrorInfo custom_op_run(const std::string& name, int32_t type, const std::string& bin_file,

const std::vector< std::string >& in_files,

const std::vector< std::string >& out_files,

const std::vector< uint32_t >& out_buf_sizes,

const std::vector< uint32_t >& workspace_sizes = std::vector<uint32_t>(),

const std::string& op_cfg_file = "", void *param = nullptr, int param_len = 0);

参数说明

参数	输入/输出	说明
name	输入	算子kernel名称
type	输入	算子类型: 0 - AICORE; 1 - AICPU
bin_file	输入	算子二进制文件路径，包含文件名
in_files	输入	输入数据文件路径，包含文件名
out_files	输入	输出数据文件路径，包含文件名
out_buf_sizes	输入	输出数据大小，单位是字节
workspace_sizes	输入	workspace大小，单位是字节

参数	输入/输出	说明
op_cfg_file	输入	用户指定的算子的输入输出描述信息
param	输入	用户指定的算子参数结构体地址信息
param_len	输入	用户指定的算子参数结构体的长度，取sizeof(结构体)

2.2 custom_op_compare 函数

函数功能

自定义算子的单算子校验接口，用于校验数据精度。

函数原型

ErrorInfo custom_op_compare(const std::string& expect_file, const std::string& actual_file,

int32_t data_type, float precision_deviation,

float statistical_discrepancy, bool& compare_result);

参数说明

参数	输入/输出	说明
expect_file	输入	期望数据文件，包含文件名
actual_file	输入	实际数据文件，包含文件名
data_type	输入	算子类型，目前支持FP32（0）、FP16（1）
precision_deviation	输入	精度偏差，单数据相对误差允许范围，数值范围（0,1），精度偏差越小表示精度越高（默认0.2）
statistical_discrepancy	输入	统计偏差, 整个数据集中精度偏差超过门限的数据量占比
compare_result	输出	比较结果， true：对比成功；false：对比失败

2.3 flip_pair 函数

函数功能

返回一个翻转之后的键值对。

函数原型

static std::pair<V, K> flip_pair(const std::pair<K, V> &p);

参数说明

参数	输入/输出	说明
p	输入	待翻转键值对

2.4 flip_map 函数

函数功能

返回翻转键值（翻转每一个键值对）对后的map。

函数原型

static std::map<V, K> flip_map(std::map<K, V> src);

参数说明

参数	输入/输出	说明
flip_map	输入	待翻转的map

2.5 AddOpAttr 函数

函数功能

将一对key-value值添加到算子的属性map中。

函数原型

void AddOpAttr(const std::string& key, AttrDef& attr, OpDef* opdef);

void AddOpAttr(const std::string& key, const std::string& value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const char* value, AttrDefMap* attrs);

void AddOpAttr(const char* key, const char* value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const uint32_t value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const int32_t value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const int64_t value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const float value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const double value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const bool value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const AttrDef_ListValue& value, AttrDefMap* attrs);

void AddOpAttr(const std::string& key, const std::string& value, OpDef* opdef);

void AddOpAttr(const std::string& key, const char* value, OpDef* opdef);

void AddOpAttr(const char* key, const char* value, OpDef* opdef);

void AddOpAttr(const std::string& key, const uint32_t value, OpDef* opdef);

void AddOpAttr(const std::string& key, const int32_t value, OpDef* opdef);

void AddOpAttr(const std::string& key, const int64_t value, OpDef* opdef);

void AddOpAttr(const std::string& key, const float value, OpDef* opdef);

void AddOpAttr(const std::string& key, const double value, OpDef* opdef);

void AddOpAttr(const std::string& key, const bool value, OpDef* opdef);

void AddOpAttr(const std::string& key, const AttrDef_ListValue& value, OpDef* opdef);

参数说明

参数	输入/输出	说明
参数1：key	输入	key值
参数2：value或attr	输入	key对应的value值
参数3：opdef或attrs	输出	opdef是需要添加键值对属性的算子 attrs是指定的属性map

2.6 AddOpBytesAttr 函数

函数功能

将bytes类型的数据添加到算子属性中。

函数原型

void AddOpBytesAttr(const std::string& key, const void * value, size_t size, OpDef* opdef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	key对应的value值
size	输入	bytes类型数据的大小，单位是字节
opdef	输出	需要添加键值对属性的算子

2.7 AddOpAttrList 函数

函数功能

将一对key-value值添加到指定的属性map的List结构中。

函数原型

void AddOpAttrList(const std::string& key, const double value, AttrDefMap* attrs);

void AddOpAttrList(const std::string& key, const float value, AttrDefMap* attrs);

void AddOpAttrList(const std::string& key, const uint32_t value, AttrDefMap* attrs);

void AddOpAttrList(const std::string& key, const int32_t value, AttrDefMap* attrs);

void AddOpAttrList(const std::string& key, const std::string value, AttrDefMap* attrs);

void AddOpAttrList(const std::string& key, const double value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const float value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const uint32_t value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const int32_t value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const bool value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const int64_t value, OpDef* opdef);

void AddOpAttrList(const std::string& key, const std::string& value, OpDef* opdef);

参数说明

参数	输入/输出	说明
参数1：key	输入	key值
参数2：value	输入	key对应的value值

参数	输入/输出	说明
参数3：attrs或opdef	输出	opdef是需要添加键值对属性的算子 attrs是指定的属性map

2.8 GetOpAttr 函数

函数功能

从算子的属性map中获取key对应的value值。

函数原型

bool GetOpAttr(const std::string& key, std::string* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, int32_t* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, int64_t* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, uint32_t* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, float* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, double* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, bool* value, const OpDef* opdef);

bool GetOpAttr(const std::string& key, AttrDef_ListValue * value, const OpDef * opdef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输出	需要获取的key对应的value值
opdef	输入	算子信息

2.9 GetOpAttrListSize 函数

函数功能

返回属性map中key对应的value（list结构）的size。

函数原型

uint32_t GetOpAttrListSize(const std::string& key, std::string value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, int32_t value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, int64_t value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, uint32_t value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, float value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, double value, const OpDef* opdef);

uint32_t GetOpAttrListSize(const std::string& key, bool value, const OpDef* opdef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	重载函数的类型标识
opdef	输入	算子信息

2.10 GetBytesAttr 函数

函数功能

从算子结构信息中根据key，获取对应的bytes类型的属性值并存到value中。

函数原型

bool GetBytesAttr(const std::string& key, std::string* value, const OpDef* opdef);

bool GetBytesAttr(const std::string& key, std::string* value, const ModelDef* modeldef);

参数说明

参数	输入/输出	说明
参数1：key	输入	key值
参数2：value	输出	需要获取的key对应的bytes结构的值
参数3：opdef或modeldef	输入	opdef指算子信息 modeldef指模型信息

2.11 AddModelAttr 函数

函数功能

将一对key-value添加到ModelDef的属性map中。

函数原型

void AddModelAttr(const std::string& key, const std::string& value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const char* value, ModelDef* modeldef);

void AddModelAttr(const char* key, const char* value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const uint32_t value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const int32_t value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const int64_t value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const float value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const double value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const bool value, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const void * value, size_t size, ModelDef* modeldef);

void AddModelAttr(const std::string& key, const AttrDef_ListValue& value, ModelDef* modeldef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	value值
modeldef	输出	ModelDef结构

函数功能

将一个key-value对中value是bytes结构的属性添加到ModelDef中。

函数原型

void AddModelAttr(const std::string& key, const void * value, size_t size, ModelDef* modeldef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	value值
size	输入	bytes的size大小，单位是字节
modeldef	输出	ModelDef结构

2.12 AddModelAttrList 函数

函数功能

将一个key-value对添加到ModelDef的属性map的list结构中。

函数原型

void AddModelAttrList(const std::string& key, const double value, ModelDef* modeldef);

void AddModelAttrList(const std::string& key, const float value, ModelDef* modeldef);

void AddModelAttrList(const std::string& key, const uint32_t value, ModelDef* modeldef);

void AddModelAttrList(const std::string& key, const int32_t value, ModelDef* modeldef);

void AddModelAttrList(const std::string& key, const std::string& value, ModelDef* modeldef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	value值
modeldef	输出	ModelDef结构

2.13 GetModelAttr 函数

函数功能

从ModelDef的属性中获取key对应的value。

函数原型

bool GetModelAttr(const std::string& key, std::string* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, int32_t* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, int64_t* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, uint32_t* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, float* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, double* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, bool* value, const ModelDef* modeldef);

bool GetModelAttr(const std::string& key, AttrDef_ListValue * value, const ModelDef* modeldef);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输出	value值
modeldef	输入	ModelDef结构

2.14 HasOpAttr 函数

函数功能

判断算子的属性中是否存在入参指定的属性，存在则返回true，否则返回false。

函数原型

bool HasOpAttr(const OpDef* opdef, const std::string attr_name);

参数说明

参数	输入/输出	说明
opdef	输入	算子结构
attr_name	输入	属性名称（key值）

2.15 SetAttrDef 函数

函数功能

在AttrDef中设置不同类型的value值

函数原型

void SetAttrDef(const std::string& value, AttrDef* out);

void SetAttrDef(const char* value, AttrDef* out);

void SetAttrDef(const uint32_t value, AttrDef* out);

void SetAttrDef(const int32_t value, AttrDef* out);

void SetAttrDef(const float value, AttrDef* out);

void SetAttrDef(const double value, AttrDef* out);

void SetAttrDef(const bool value, AttrDef* out);

参数说明

参数	输入/输出	说明
value	输入	需要保存到AttrDef中的value值
out	输出	AttrDef结构

2.16 SetAttrList 函数

函数功能

在AttrDef的list结构中设置不同类型的value值。

函数原型

void SetAttrList(const std::string& value, AttrDef* out);

void SetAttrList(const bool value, AttrDef* out);

void SetAttrList(const float value, AttrDef* out);

void SetAttrList(const double value, AttrDef* out);

void SetAttrList(const uint32_t value, AttrDef* out);

参数说明

参数	输入/输出	说明
value	输入	需要保存到AttrDef的list中的value值
out	输出	AttrDef结构

2.17 GetAttrDefValue 函数

函数功能

从AttrDefMap中获取key对应的value值。

函数原型

bool GetAttrDefValue(const std::string& key, std::string* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, int32_t* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, int64_t* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, uint32_t* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, float* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, double* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, bool* value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, AttrDef_ListValue * value, const AttrDefMap &attr);

bool GetAttrDefValue(const std::string& key, NamedAttrs *& value, AttrDefMap *attr);

bool GetAttrDefValue(const std::string& key, const NamedAttrs *& value, const AttrDefMap &attr);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输出	获取的value值
attr	输入	属性map

2.18 GetAttrDefListValue 函数

函数功能

从AttrDefMap的list结构中获取key对应的value值。

函数原型

bool GetAttrDefListValue(const std::string& key, int idx, int32_t* value, const AttrDefMap &attr);

bool GetAttrDefListValue(const std::string& key, int idx, uint32_t* value, const AttrDefMap &attr);

bool GetAttrDefListValue(const std::string& key, int idx, float* value, const AttrDefMap &attr);

bool GetAttrDefListValue(const std::string& key, int idx, double* value, const AttrDefMap &attr);

参数说明

参数	输入/输出	说明
key	输入	key值
idx	输入	list结构中的索引
value	输出	获取的value值
attr	输入	属性map

2.19 ConvertToOpDef 函数

函数功能

将Operator中的信息转为OpDef的结构。

函数原型

Status ConvertToOpDef(const Operator& op, OpDef* op_def);

参数说明

参数	输入/输出	说明
op	输入	Operator结构信息

参数	输入/输出	说明
op_def	输出	转换后的OpDef结构

2.20 ConvertFromOpDef 函数

函数功能

将OpDef中的信息转为Operator的结构。

函数原型

Status ConvertFromOpDef(const OpDef* op_def, Operator& op);

参数说明

参数	输入/输出	说明
op_def	输入	OpDef结构信息
op	输出	转换后的Operator结构信息

2.21 GetContext 函数

函数功能

获取OMG上下文，并返回。

函数原型

OmgContext& GetContext();

参数说明

无。

3 类成员函数

3.1 Graph 类

3.1.1 Graph 函数

函数功能

构造函数。

函数原型

explicit Graph();

参数说明

无。

3.1.2 ~Graph 函数

函数功能

析构函数。

函数原型

virtual ~Graph();

参数说明

无。

3.1.3 AddNode 函数

函数功能

往Graph内添加节点，并把输入的op_def填充到节点中，返回当前节点信息。

函数原型

Node* AddNode(OpDef* op_def);

参数说明

参数	输入/输出	说明
op_def	输入	用于填充节点op_def信息

3.1.4 AddNode 函数（无返回值）

函数功能

往Graph内添加节点，并把输入的op_def填充到节点中，输出以形参返回。

函数原型

void AddNode(Node *node,OpDef *op_def);

参数说明

参数	输入/输出	说明
node	输出	返回的node节点信息
op_def	输入	用于填充节点op_def信息

3.1.5 RemoveNode 函数

函数功能

从Graph中移除指定节点，成功返回SUCCESS，否则返回非SUCCESS。

函数原型

Status RemoveNode(Node* node);

参数说明

参数	输入/输出	说明
node	输入	待移除节点的指针

3.1.6 IsolateNode 函数

函数功能

将节点孤立，即将该节点的前驱节点与后继节点相连，并将该节点的边关系删除。仅能处理输入边数小于等于1、输出边数大于等于1的场景。

函数原型

Status IsolateNode(Node* node);

参数说明

参数	输入/输出	说明
node	输入	待孤立节点

3.1.7 IsolateMulInutNode 函数

函数功能

将节点孤立，即将该节点的前驱节点与后继节点相连，并将该节点的边关系删除，处理多输入边场景。

函数原型

Status IsolateMulInutNode(Node* node);

参数说明

参数	输入/输出	说明
node	输入	待孤立节点

3.1.8 MergeNodeEdge 函数

函数功能

将节点的多个输入边合并。

函数原型

Status MergeNodeEdge(Node* node);

参数说明

参数	输入/输出	说明
node	输入	待合并输入边的节点

3.1.9 AddEdge 函数（含 format 类型）

函数功能

为两个有依赖关系的节点添加边。

函数原型

Edge* AddEdge(Node* src, int32_t x, int32_t src_format, Node* dest, int32_t y, int32_t dst_format, bool ctrl = false);

参数说明

参数	输入/输出	说明
src	输入	边的源节点
x	输入	源节点的输出索引
src_format	输入	源节点的format类型
dest	输入	边的目的节点
y	输入	目标节点的输入索引
dst_format	输入	目标节点的format类型
ctrl	输入	控制边属性

3.1.10 AddEdge 函数（不含 format 类型）

函数功能

为两个有依赖关系的节点添加边。

函数原型

Edge* AddEdge(Node* src, int32_t x, Node* dest, int32_t y, bool ctrl = false);

参数说明

参数	输入/输出	说明
src	输入	边的源节点
x	输入	源节点的输出索引
dest	输入	边的目的节点
y	输入	目标节点的输入索引
ctrl	输入	控制边属性

3.1.11 RemoveEdge 函数

函数功能

从Graph中移除指定边。

函数原型

Status RemoveEdge(Edge* edge, bool reindex = true);

参数说明

参数	输入/输出	说明
edge	输入	待移除边的指针
reindex	输入	是否需要对边重新编号，为了对边进行替换，提供这个参数，默认为true

3.1.12 FindNode 函数

函数功能

根据节点的名称查找节点。

函数原型

Node* FindNode(const std::string name);

参数说明

参数	输入/输出	说明
name	输入	节点的名称

3.1.13 GetAllNodes 函数

函数功能

获取该Graph的所有节点。

函数原型

const std::vector<Node*>& GetAllNodes();

参数说明

无。

3.1.14 GetAllEdges 函数

函数功能

获取该Graph的所有边。

函数原型

const std::vector<Edge*>& GetAllEdges();

参数说明

无。

3.1.15 SetName 函数

函数功能

设置graph名称。

函数原型

void SetName(const std::string name);

参数说明

参数	输入/输出	说明
name	输入	设置的graph名称

3.1.16 Name 函数

函数功能

获取graph名称。

函数原型

const std::string& Name();

参数说明

无。

3.1.17 TopologicalSorting 函数

函数功能

对该Graph进行拓扑排序

函数原型

Status TopologicalSorting();

参数说明

无。

3.1.18 IsValid 函数

函数功能

判断该Graph是否合法(包括是否有环、是否有孤立子图、拓扑排序和节点个数)，合法，返回true，不合法，返回false。

函数原型

bool IsValid();

参数说明

无。

3.1.19 Print 函数

函数功能

打印整个graph。

函数原型

void Print();

参数说明

无。

3.1.20 InsertNodeInEdge 函数

函数功能

在一条边上插入一个节点。

函数原型

Status InsertNodeInEdge(Edge* e, std::unique_ptr<OpDef> op_def);

参数说明

参数	输入/输出	说明
e	输入	待插入节点的边
op_def	输入	待插入的节点

函数功能

在一条边上插入多个节点。

函数原型

Status InsertNodeInEdge(Edge* e, std::vector<OpDef*>& vec_op_def);

参数说明

参数	输入/输出	说明
e	输入	待插入节点的边
vec_op_def	输入	待插入的多个节点

函数功能

在一条边上插入一个节点。

函数原型

Status InsertNodeInEdge(Edge* e, Node* node);

参数说明

参数	输入/输出	说明
e	输入	待插入节点的边
node	输入	待插入的节点

3.1.21 GetShareParamLayer()函数

函数功能

获取该Graph的所有共享权值的layer集合

函数原型

const std::map<std::vector<std::string>, std::vector<std::string>>& GetShareParamLayer();

参数说明

无。

3.1.22 SetShareParamLayer 函数

函数功能

设置该Graph的所有共享权值的layer集合。

函数原型

void SetShareParamLayer(const std::map<std::vector<std::string>, std::vector<std::string>> params_share_map);

参数说明

参数	输入/输出	说明
params_share_map	输入	共享权值的layer集合

3.1.23 SortNodes 函数

函数功能

为节点排序。

函数原型

Status SortNodes(std::vector<Node*> & stack,std::vector<uint32_t>& in_edge_num);

参数说明

参数	输入/输出	说明
stack	输入	存放节点
in_edge_num	输入	输入边的个数

3.1.24 IsValidNode 函数

函数功能

判断指定的节点是否合法，合法返回true，否则返回false。

函数原型

bool IsValidNode(Node* node);

参数说明

参数	输入/输出	说明
node	输入	指定的节点

3.2 Node 类

3.2.1 Node 函数

函数功能

构造函数。

函数原型

Node();

参数说明

无。

3.2.2 ~Node 函数

函数功能

析构函数。

函数原型

virtual ~Node();

参数说明

无。

3.2.3 Name 函数

函数功能

获取节点名称。

函数原型

const std::string& Name() const;

参数说明

无。

3.2.4 Id 函数

函数功能

获取节点ID。

函数原型

uint32_t Id();

参数说明

无。

3.2.5 SetId 函数

函数功能

设置节点ID。

函数原型

void SetId(uint32_t new_id);

参数说明

参数	输入/输出	说明
new_id	输入	节点ID

3.2.6 OutEdges 函数

函数功能

获取该节点的输出边全集。

函数原型

const std::vector<Edge*>& OutEdges();

参数说明

无。

3.2.7 OutControlEdges 函数

函数功能

获取该节点的输出控制边。

函数原型

const std::vector<Edge*>& OutControlEdges();

参数说明

无。

3.2.8 OutNormalEdges 函数

函数功能

获取该节点的输出普通边。

函数原型

const std::vector<Edge*>& OutNormalEdges();

参数说明

无。

3.2.9 InEdges 函数

函数功能

获取该节点的输入边全集。

函数原型

const std::vector<Edge*>& InEdges() const;

参数说明

无。

3.2.10 InControlEdges 函数

函数功能

获取该节点的输入控制边全集。

函数原型

const std::vector<Edge*>& InControlEdges();

参数说明

无。

3.2.11 InNormalEdges 函数

函数功能

获取该节点的输入普通边全集。

函数原型

const std::vector<Edge*>& InNormalEdges();

参数说明

无。

3.2.12 GetOutputTensors 函数

函数功能

获取该节点的输出Tensor(tensorDescriptor_+内存地址)。

函数原型

const std::vector<calibration::Tensor*>& GetOutputTensors();

参数说明

无。

3.2.13 SetOutputTensors 函数

函数功能

设置该节点的输出Tensor(tensorDescriptor_+内存地址)。

函数原型

void SetOutputTensors(std::vector<calibration::Tensor*> output_tensors);

参数说明

参数	输入/输出	说明
output_tensors	输入	输出Tensor值

3.2.14 GetInputTensors 函数

函数功能

获取该节点的输入Tensor(tensorDescriptor_+内存地址)。

函数原型

const std::vector<calibration::Tensor*>& GetInputTensors();

参数说明

无。

3.2.15 SetInputTensors 函数

函数功能

设置该节点的输入Tensor(tensorDescriptor_+内存地址)。

函数原型

void SetInputTensors(std::vector<calibration::Tensor*> input_tensors);

参数说明

参数	输入/输出	说明
input_tensors	输入	输入Tensor

3.2.16 SetGraph 函数

函数功能

设置该节点所在的Graph。

函数原型

void SetGraph(Graph* graph);

参数说明

参数	输入/输出	说明
graph	输入	该节点所在的Graph

3.2.17 GetGraph 函数

函数功能

获取该节点的Graph信息。

函数原型

const Graph* GetGraph() const;

参数说明

无。

函数功能

获取该节点的Graph信息。

函数原型

Graph* GetGraph();

参数说明

无。

3.2.18 GetOpDef 函数

函数功能

获取该节点的op_def信息。

函数原型

const OpDef* GetOpDef() const;

参数说明

无。

函数功能

获取该节点的op_def信息。

函数原型

OpDef* GetOpDef();

参数说明

无

3.2.19 GetConstInputs 函数

函数功能

获取const输入节点。

函数原型

std::vector<Node*> GetConstInputs();

参数说明

无

3.2.20 RemoveInEdge 函数

函数功能

从Node的InEdge中移除指定边。

函数原型

void RemoveInEdge(Edge* edge, bool reindex = true);

参数说明

参数	输入/输出	说明
edge	输入	待移除边的指针
reindex	输入	是否对剩下的边进行重新编号，默认为true

3.2.21 RemoveOutEdge 函数

函数功能

从Node的OutEdge中移除指定边。

函数原型

void RemoveOutEdge(Edge* edge, bool reindex = true);

参数说明

参数	输入/输出	说明
edge	输入	待移除边的指针
reindex	输入	是否对剩下的边进行重新编号，默认为true

3.2.22 RemoveControlOrNormalEdge 函数

函数功能

删除控制或普通边。

函数原型

void RemoveControlOrNormalEdge(std::vector<Edge*>& control_edges, std::vector<Edge*>& normal_edges, Edge* edge);

参数说明

参数	输入/输出	说明
control_edges	输入	控制边集合
normal_edges	输入	普通边集合
edge	输入	待移除边的指针

3.2.23 AddSendEventId 函数

函数功能

向发送事件列表中新增指定的event_id。

函数原型

void AddSendEventId(uint32_t event_id);

参数说明

参数	输入/输出	说明
event_id	输入	事件id

3.2.24 AddRecvEventId 函数

函数功能

向接收事件列表中新增指定的event_id。

函数原型

void AddRecvEventId(uint32_t event_id);

参数说明

参数	输入/输出	说明
event_id	输入	事件id

3.2.25 GetSendEventIdList 函数

函数功能

获取发送事件列表。

函数原型

const std::vector<uint32_t>& GetSendEventIdList();

参数说明

无。

3.2.26 GetRecvEventIdList 函数

函数功能

获取接收事件列表。

函数原型

const std::vector<uint32_t>& GetRecvEventIdList();

参数说明

无。

3.2.27 operator=函数

函数功能

赋值构造函数。

函数原型

Node& operator=(const Node &node);

参数说明

参数	输入/输出	说明
node	输入	该对象的引用

3.2.28 Node 函数（含入参）

函数功能

拷贝构造函数。

函数原型

Node(const Node &node);

参数说明

参数	输入/输出	说明
node	输入	该对象的引用

3.3 Edge 类

3.3.1 Edge 函数

函数功能

构造函数。

函数原型

Edge();

参数说明

无。

3.3.2 ~Edge 函数

函数功能

析构函数。

函数原型

~Edge();

参数说明

无。

3.3.3 Src 函数

函数功能

获取边的源节点指针。

函数原型

Node* Src() const;

参数说明

无。

3.3.4 Dst 函数

函数功能

获取边的目标节点指针。

函数原型

Node* Dst() const;

参数说明

无

3.3.5 SrcIndex 函数

函数功能

获取边的源节点索引。

函数原型

int SrcIndex();

参数说明

无。

3.3.6 DstIndex 函数

函数功能

获取边的目标节点索引。

函数原型

int DstIndex();

参数说明

无。

3.3.7 SetDstIndex 函数

函数功能

设置边的目标节点索引。

函数原型

void SetDstIndex(int32_t new_index);

参数说明

参数	输入/输出	说明
new_index	输入	目标节点索引

3.3.8 IsControl 函数

函数功能

获取边属性是否为控制边，是，返回true，否，返回false。

函数原型

int IsControl();

参数说明

无。

3.3.9 SetControl 函数

函数功能

设置边属性是否为控制边。

函数原型

void SetControl(bool ctrl);

参数说明

参数	输入/输出	说明
Ctrl	输入	边属性是否为控制边，是控制边值为true，不是控制边值为false

3.3.10 GetSwtichOutIndex 函数

函数功能

获取Switch Node的输出真实index。

函数原型

int32_t GetSwtichOutIndex();

参数说明

无。

3.3.11 SrcFormat 函数

函数功能

获取边的源节点format。

函数原型

int32_t SrcFormat();

参数说明

无。

3.3.12 SetSrcFormat 函数

函数功能

设置边的源节点format。

函数原型

void SetSrcFormat(int32_t src_format);

参数说明

参数	输入/输出	说明
src_format	输入	源节点的format

3.3.13 DstFormat 函数

函数功能

获取边的目标节点format。

函数原型

int32_t DstFormat();

参数说明

无。

3.3.14 SetDstFormat 函数

函数功能

设置边的目标节点format。

函数原型

void SetDstFormat(int32_t dst_format);

参数说明

参数	输入/输出	说明
dst_format	输入	目标节点format

3.4 StatusFactory 类

3.4.1 Instance 函数

函数功能

返回Status工厂类的实例对象。

函数原型

static StatusFactory* Instance();

参数说明

无

3.4.2 RegisterErrorNo 函数

函数功能

注册错误码。

函数原型

void RegisterErrorNo(uint32_t err, const std::string& desc);

参数说明

参数	输入/输出	说明
err	输入	错误码
desc	输入	错误码描述字符串

3.4.3 GetErrDesc 函数

函数功能

获取错误码描述。

函数原型

std::string GetErrDesc(uint32_t err);

参数说明

参数	输入/输出	说明
err	输入	错误码

3.4.4 StatusFactory 函数

函数功能

构造函数。

函数原型

StatusFactory();

参数说明

无

3.4.5 ~StatusFactory 函数

函数功能

析构函数。

函数原型

~StatusFactory();

参数说明

无

3.5 ErrorNoRegisterar 类

3.5.1 ErrorNoRegisterar 函数

函数功能

构造函数。

函数原型

ErrorNoRegisterar(uint32_t err, const std::string& desc);

参数说明

参数	输入/输出	说明
err	输入	错误码
desc	输入	错误码描述字符串

3.5.2 ~ErrorNoRegisterar 函数

函数功能

析构函数。

函数原型

~ErrorNoRegisterar();

参数说明

无。

3.6 OpTypeContainer 类

3.6.1 Instance 函数

函数功能

获取OpTypeContainer工厂类的实例对象。

函数原型

static OpTypeContainer * Instance();

参数说明

无。

3.6.2 Register 函数

函数功能

将算子的类型保存到一个set中。

函数原型

void Register(const std::string& op_type);

参数说明

参数	输入/输出	说明
op_type	输入	需要保存的算子类型

3.6.3 IsExisting 函数

函数功能

判断一个算子类型是否存在。

函数原型

bool IsExisting(const std::string& op_type);

参数说明

参数	输入/输出	说明
op_type	输入	需要查询的算子类型

3.6.4 OpTypeContainer 函数

函数功能

构造函数。

函数原型

OpTypeContainer();

参数说明

无。

3.7 OpTypeRegistrar 类

3.7.1 OpTypeRegistrar 函数

函数功能

构造函数，将一个算子的类型注册到Container中。

函数原型

OpTypeRegistrar(const std::string& op_type);

参数说明

参数	输入/输出	说明
op_type	输入	需要注册的算子类型

3.7.2 ~OpTypeRegistrar 函数

函数功能

析构函数。

函数原型

~OpTypeRegistrar();

参数说明

无

3.8 Operator 类

3.8.1 Operator 函数（带参数）

函数功能

构造函数，注册一种算子类型。

函数原型

Operator(const std::string& type);

参数说明

参数	输入/输出	说明
type	输入	算子类型名称

3.8.2 Operator 函数（不带参数）

函数功能

构造函数。

函数原型

Operator();

参数说明

无

3.8.3 ~Operator 函数

函数功能

析构函数。

函数原型

virtual ~Operator();

参数说明

无。

3.8.4 Input 函数

函数功能

将输入op的名称加上index索引，以一个字符串的形式存放到当前op中，并返回当前op。

函数原型

Operator& Input(const Operator& in_op, uint32_t index = 0);

参数说明

参数	输入/输出	说明
in_op	输入	输入op
index	输入	索引

3.8.5 Attr 函数

函数功能

在当前op的属性map中寻找是否有输入的op_attr，若有则覆盖更新，否则新增该attr。

函数原型

Operator& Attr(const OpAttribute& op_attr);

Operator& Attr(const std::string& name, const TypeName& value);

Operator& Attr(const std::string& name, const std::vector< TypeName >& value);

Operator& Attr(const std::string& name, const domi::Tuple< TypeName >& value);

参数说明

参数	输入/输出	说明
参数1：op_attr 或者 name	输入	OpAttribute指需要增加的属性 name对应key值
参数2：value	输入	value是name对应的值，可以是uint32_t、int64_t、bool、float、std::string、std::vector、domi::Tuple类型

3.8.6 AttrVector 函数

函数功能

根据key判断当前op的属性map中是否存在一个OpAttribute，然后使用key和value构造出的OpAttribute来更新或者新增一个OpAttribute。

函数原型

Operator& AttrVector(std::string key, std::vector< int32_t >& value);

参数说明

参数	输入/输出	说明
key	输入	key值
value	输入	value值

3.8.7 Name 函数

函数功能

设置当前op的名称。

函数原型

Operator& Name(const std::string& name);

参数说明

参数	输入/输出	说明
name	输入	需要设置的名称

3.8.8 Type 函数

函数功能

设置当前op的类型。

函数原型

Operator& Type(const std::string& type);

参数说明

参数	输入/输出	说明
type	输入	需要设置的类型名称

3.8.9 InputTensorDesc 函数

函数功能

设置输入tensor描述。

函数原型

Operator& InputTensorDesc(const TensorDescriptor& input_tensordesc);

参数说明

参数	输入/输出	说明
input_tensordesc	输入	需要设置的TensorDescriptor变量

3.8.10 OutputTensorDesc 函数

函数功能

用设置输出tensor描述。

函数原型

Operator& OutputTensorDesc(const TensorDescriptor& output_tensordesc);

参数说明

参数	输入/输出	说明
output_tensordesc	输入	需要设置的TensorDescriptor变量

3.8.11 Attr_bt 函数

函数功能

设置bytes类型的数据到op的属性中。

函数原型

Operator& Attr_bt(const std::string& name, const std::string& value);

参数说明

参数	输入/输出	说明
name	输入	属性对应的key值
value	输入	属性对应的value值

3.8.12 Get 函数

函数功能

返回Operator的成员变量。

函数原型

const std::string& GetName() const;

const std::string& GetType() const;

const std::vector< std::string >& GetInputs() const;


```
const std::vector< TensorDescriptor >& GetInputTensorDesc() const;
const std::vector< TensorDescriptor >& GetOutputTensorDesc() const;
const std::unordered_map< std::string, OpAttribute > GetOpAttrs() const;
const OpSchema* GetSchema() const;
```

参数说明

无。

3.8.13 GetAttr 函数

函数功能

返回op的属性map中，各种类型数据的属性值。

函数原型

```
int64_t GetIntAttr(const std::string& name) const;
uint32_t GetUintAttr(const std::string& name) const;
float GetFloatAttr(const std::string& name) const;
bool GetBoolAttr(const std::string& name) const;
std::string GetStringAttr(const std::string& name) const;
IntTuple GetIntTupleAttr(const std::string& name) const;
UintTuple GetUintTupleAttr(const std::string& name) const;
FloatTuple GetFloatTupleAttr(const std::string& name) const;
BoolTuple GetBoolTupleAttr(const std::string& name) const;
StringTuple GetStringTupleAttr(const std::string& name) const;
```

参数说明

参数	输入/输出	说明
name	输入	属性map对应的key值

3.8.14 HasAttr 函数

函数功能

判断op的map中是否存在这个attr，存在则返回true，否则返回false。

函数原型

bool HasAttr(const std::string& name) const;

参数说明

参数	输入/输出	说明
name	输入	属性map对应的key值

3.9 OpSchema 类

3.9.1 OpSchema 函数

函数功能

构造函数。

函数原型

OpSchema(const std::string& name);

参数说明

参数	输入/输出	说明
name	输入	设置op模式的名称

3.9.2 Input 函数

函数功能

设置名称为name的入参模式。

函数原型

OpSchema& Input(const std::string& name, FormalParameterOption param_option = Single);

参数说明

参数	输入/输出	说明
name	输入	入参名称

参数	输入/输出	说明
param_option	输入	入参模式（默认Single：1个，Optional：0个或1个，Variadic：1个或多个）

3.9.3 Output 函数

函数功能

设置名称为name的出参模式。

函数原型

OpSchema& Output(const std::string& name, FormalParameterOption param_option = Single);

参数说明

参数	输入/输出	说明
name	输入	出参名称
param_option	输入	出参模式（默认Single：1个，Optional：0个或1个，Variadic：1个或多个）

3.9.4 Attr 函数

函数功能

为当前OpShema添加一种属性。

函数原型

OpSchema& Attr(const Attribute& attr);

OpSchema& Attr(const std::string& name, AttributeType type, const TypeName& defaultValue);

OpSchema& Attr(const std::string& name, AttributeType type,

const std::vector<TypeName>& defaultValue);

OpSchema& Attr(const std::string& name, AttributeType type,

const domi::Tuple<TypeName>& defaultValue);

参数说明

参数	输入/输出	说明
参数1：attr 或者 name	输入	attr 指需要添加的属性 name 指key值
参数2：type	输入	type指属性类型
参数3：defaultValue	输入	设置的值，可以是uint32_t、int64_t、bool、float、std::string、std::vector、domi::Tuple类型

3.9.5 AttrRequired 函数

函数功能

添加一个必须的数据类型属性。

函数原型

OpSchema& AttrRequired(const std::string& name, AttributeType type);

参数说明

参数	输入/输出	说明
name	输入	属性名称
type	输入	该属性的数据类型

3.9.6 HasDefaultAttr 函数

函数功能

判断该属性是否有默认值，有则返回true，否则返回false。

函数原型

bool HasDefaultAttr(const std::string& name) const;

参数说明

参数	输入/输出	说明
name	输入	属性名称

3.9.7 GetDefaultAttr 函数

函数功能

获取默认属性值。

函数原型

const AttrDef& GetDefaultAttr(const std::string& name) const;

参数说明

参数	输入/输出	说明
name	输入	属性名称

3.9.8 Verify 函数

函数功能

校验OpDef是否符合要求。

函数原型

bool Verify(const OpDef* op_def) const;

参数说明

参数	输入/输出	说明
op_def	输入	算子结构

3.10 FormalParameter 类

3.10.1 FormalParameter 函数

函数功能

构造函数。

函数原型

explicit FormalParameter(const std::string& name, FormalParameterOption param_option = Single);

FormalParameter();

参数说明

参数	输入/输出	说明
name	输入	参数名称
param_option	输入	参数模式（默认Single：1个，Optional：0个或1个，Variadic：1个或多个）

3.10.2 Name 函数

函数功能

获取参数名称。

函数原型

const std::string& Name() const;

参数说明

无。

3.10.3 Option 函数

函数功能

获取参数模式。

函数原型

FormalParameterOption Option() const;

参数说明

无。

3.11 OpSchemaFactory 类

3.11.1 Instance 函数

函数功能

获取模式工厂的实例。

函数原型

static OpSchemaFactory* Instance();

参数说明

无。

3.11.2 Get 函数

函数功能

返op对应的OpSchema。

函数原型

const OpSchema* Get(const std::string& op) const;

参数说明

参数	输入/输出	说明
op	输入	op的名称

3.11.3 OpSchemaFactory 函数

函数功能

构造函数。

函数原型

OpSchemaFactory();

参数说明

无

3.11.4 ~OpSchemaFactory 函数

函数功能

析构函数。

函数原型

~OpSchemaFactory();

参数说明

无

3.12 OpSchemaRegistry 类

3.12.1 OpSchemaRegistry 函数

函数功能

构造函数，注册一种OpSchema。

函数原型

OpSchemaRegistry(OpSchema& op_schema);

参数说明

参数	输入/输出	说明
op_schema	输入	需要注册的OpSchema

3.12.2 ~OpSchemaRegistry 函数

函数功能

析构函数。

函数原型

~OpSchemaRegistry();

参数说明

无。

3.13 OpRegistrationData 类

3.13.1 OpRegistrationData 函数

函数功能

构造函数。

函数原型

OpRegistrationData(const std::string& om_optype);

参数说明

参数	输入/输出	说明
om_optype	输入	算子类型

3.13.2 FrameworkType 函数

函数功能

设置框架类型。

函数原型

OpRegistrationData& FrameworkType(domi::FrameworkType fmk_type);

参数说明

参数	输入/输出	说明
fmk_type	输入	框架类型 0：caffe 3：tensorflow

3.13.3 OriginOpType 函数

函数功能

设置原始模型的算子类型列表。

函数原型

OpRegistrationData& OriginOpType (

const std::initializer_list<std::string> & ori_optype_list);

参数说明

参数	输入/输出	说明
ori_optype_list	输入	原始模型算子类型列表

3.13.4 OriginOpType 函数

函数功能

设置原始模型的算子类型。

函数原型

OpRegistrationData& OriginOpType (const std::string& ori_optype);

参数说明

参数	输入/输出	说明
ori_optype	输入	原始模型算子类型

3.13.5 ParserCreatorFn 函数

函数功能

设置算子的parser创建函数。

函数原型

OpRegistrationData& ParserCreatorFn(PARSER_CREATOR_FN parser_creator_fn);

参数说明

参数	输入/输出	说明
parser_creator_fn	输入	Parser创建函数指针

3.13.6 BuilderCreatorFn 函数

函数功能

设置算子的Builder创建函数。

函数原型

OpRegistrationData& BuilderCreatorFn(BUILDER_CREATOR_FN builder_creator_fn);

参数说明

参数	输入/输出	说明
builder_creator_fn	输入	Builder创建函数指针

3.13.7 ImplyType 函数

函数功能

设置算子执行方式。

函数原型

OpRegistrationData& ImplyType(domi::ImplyType imply_type);

参数说明

参数	输入/输出	说明
imply_type	输入	算子执行方式

3.13.8 Formats 函数

函数功能

设置算子支持的输入/输出数据格式。

函数原型

OpRegistrationData& Formats(

const std::initializer_list<domi::tagDomiTensorFormat>& input_formats,

const std::initializer_list<domi::tagDomiTensorFormat>& output_formats);

OpRegistrationData& Formats(

const domi::tagDomiTensorFormat& input_format,

const domi::tagDomiTensorFormat& output_format);

参数说明

参数	输入/输出	说明
input_format	输入	输入支持的数据格式
output_format	输入	输出支持的数据格式

3.13.9 WeightFormats 函数

函数功能

设置算子权重支持的数据格式。

函数原型

OpRegistrationData& WeightFormats(

const std::initializer_list<domi::tagDomiTensorFormat>& weight_formats);

参数说明

参数	输入/输出	说明
weight_formats	输入	权重支持的数据格式列表

3.13.10 Finalize 函数

函数功能

创建所有支持的算子的parser和builder函数，成功则返回true，否则返回false。

函数原型

bool Finalize();

参数说明

无。

3.14 OpRegistry 类

3.14.1 Instance 函数

函数功能

获取OpRegistry类对象的实例。

函数原型

static OpRegistry* Instance();

参数说明

无。

3.14.2 Register 函数

函数功能

注册一个OpRegistrationData，成功则返回true，否则返回false。

函数原型

bool Register(const OpRegistrationData& reg_data);

参数说明

参数	输入/输出	说明
reg_data	输入	算子注册信息

3.14.3 GetImplyType 函数

函数功能

获取算子执行类型。

函数原型

domi::ImplyType GetImplyType(const std::string& op_type);

参数说明

参数	输入/输出	说明
op_type	输入	需要查询的算子类型

3.14.4 GetOpTypeByImplyType 函数

函数功能

查询指定执行类型的所有算子的名称。

函数原型

void GetOpTypeByImplyType(std::vector<std::string>& vec_op_type,const domi::ImplyType& imply_type);

参数说明

参数	输入/输出	说明
vec_op_type	输出	保存查询的算子名称信息
imply_type	输入	算子的执行类型

3.14.5 GetFormats 函数

函数功能

查询op_type这个算子支持的输入输出数据类型。

函数原型

void GetFormats(const std::string& op_type, std::vector<domi::tagDomiTensorFormat>& input_format_vector,std::vector<domi::tagDomiTensorFormat>& output_format_vector);

参数说明

参数	输入/输出	说明
op_type	输入	算子类型名称
input_format_vector	输出	支持的输入数据类型列表

参数	输入/输出	说明
output_format_vector	输出	支持的输出数据类型列表

3.14.6 GetWeightFormats 函数

函数功能

查询op_type这个算子支持的权重数据类型。

函数原型

void GetWeightFormats(const std::string& op_type,

std::vector<domi::tagDomiTensorFormat>& format_vector);

参数说明

参数	输入/输出	说明
op_type	输入	算子类型名称
format_vector	输出	支持的权重数据类型列表

3.15 OpReceiver 类

3.15.1 OpReceiver 函数

函数功能

构造函数。

函数原型

OpReceiver(OpRegistrationData& reg_data);

参数说明

参数	输入/输出	说明
reg_data	输入	需要注册的算子信息

3.15.2 ~OpReceiver 函数

函数功能

析构函数。

函数原型

~OpReceiver();

参数说明

无。

3.16 OpBuilder 类

3.16.1 OpBuilder 函数

函数功能

构造函数，采用一个算子结构初始化对象。

函数原型

OpBuilder(OpDef* op_def);

参数说明

参数	输入/输出	说明
op_def	输入	算子结构

3.16.2 ~OpBuilder 函数

函数功能

析构函数。

函数原型

virtual ~OpBuilder();

参数说明

无。

3.16.3 SetNode 函数

函数功能

设置一个节点（算子对应的图节点）。

函数原型

void SetNode(Node* node);

参数说明

参数	输入/输出	说明
node	输入	算子对应的图节点

3.16.4 Build 函数

函数功能

执行算子的build操作。

函数原型

Status Build(size_t& mem_offset);

参数说明

参数	输入/输出	说明
mem_offset	输入	内存偏移

3.16.5 GetOutputDesc 函数

函数功能

获取算子的输出描述，需要在子类中实现。

函数原型

virtual Status GetOutputDesc(vector<TensorDescriptor>& v_output_desc);

参数说明

参数	输入/输出	说明
v_output_desc	输入	输出描述

3.16.6 AdjustParams 函数

函数功能

调整算子的模型定义，需要在子类中实现。

函数原型

virtual Status AdjustParams();

参数说明

无。

3.16.7 GetWorkSpaceMemory 函数

函数功能

获取算子的所有workspace内存大小，需要在子类中实现。

函数原型

virtual Status GetWorkSpaceMemory(vector<int64_t>& v_workspace_memory);

参数说明

参数	输入/输出	说明
v_workspace_memory	输入	内存值，单位是字节

3.16.8 TransWeights 函数

函数功能

转换算子内所有权重数据的数据类型并计算内存偏移，需要在子类中实现。

函数原型

virtual Status TransWeights(size_t& mem_offset);

参数说明

参数	输入/输出	说明
mem_offset	输出	内存偏移

3.16.9 TransWeight 函数

函数功能

转换算子内所有权重数据的数据类型并计算内存偏移，需要在子类中实现。

函数原型

static Status TransWeight(const cce::ccTensorDescriptor_t tensor_desc,

WeightDef* weight, size_t& mem_offset);

参数说明

参数	输入/输出	说明
tensor_desc	输入	待转换权重数据的描述
weight	输出	待转换的权重信息
mem_offset	输出	内存偏移

3.16.10 SetFormatAndDatatype 函数

函数功能

设置算子的format和datatype。

函数原型

virtual void SetFormatAndDatatype(vector<TensorDescriptor>& v_output_desc);

参数说明

参数	输入/输出	说明
v_output_desc	输出	设置后的描述信息

3.16.11 BuildTvmBinFile 函数

函数功能

设置TVM这类算子的bin信息，需要在子类中实现。

函数原型

virtual Status BuildTvmBinFile(TVMBinInfo& tvm_bin_info);

参数说明

参数	输入/输出	说明
tvm_bin_info	输入	tvm二进制信息

3.16.12 IsBuildinOp 函数

函数功能

判断是否是框架内置算子类型，如果是返回true，否则返回false。

函数原型

static bool IsBuildinOp(OpDef*& op_def);

参数说明

参数	输入/输出	说明
op_def	输入	算子结构

3.16.13 SetQuantizeFactorOffset 函数

函数功能

设置算子的量化因子偏移。

函数原型

static Status SetQuantizeFactorOffset(QuantizeFactor* quantize_factor,

size_t& mem_offset);

参数说明

参数	输入/输出	说明
quantize_factor	输出	量化因子
mem_offset	输出	偏移量

3.16.14 SetQuantizeFactorParamsOffset 函数

函数功能

设置算子的量化因子中各参数的偏移。

函数原型

static Status SetQuantizeFactorParamsOffset(QuantizeFactorParams* quantize_factor_params, size_t& mem_offset);

参数说明

参数	输入/输出	说明
quantize_factor_params	输出	量化因子参数
mem_offset	输出	偏移量

3.16.15 GetImgInfo 函数

函数功能

获取算子WeightDef中高和宽。

函数原型

static Status GetImgInfo(int32_t &img_h,int32_t &img_w,OpDef*& op_def);

参数说明

参数	输入/输出	说明
img_h	输出	高
img_w	输出	宽

参数	输入/输出	说明
op_def	输入	算子结构

3.16.16 SetRealDimCnt 函数

函数功能

设置realdimcnt列表。

函数原型

virtual Status SetRealDimCnt(vector<uint32_t>);

参数说明

参数	输入/输出	说明
vector<uint32_t>	输入	设置列表

3.16.17 GetImgInfoFromInput 函数

函数功能

从用户指定的输入中解析原始图片的高和宽。

函数原型

static Status GetImgInfoFromInput(int32_t &img_h,int32_t &img_w);

参数说明

参数	输入/输出	说明
img_h	输出	高
img_w	输出	宽

3.16.18 Init 函数

函数功能

初始化Op Builder，需要在子类中实现。

函数原型

virtual Status Init();

参数说明

无。

3.16.19 BuildTvmOp 函数

函数功能

构建TVM的Op。

函数原型

virtual Status BuildTvmOp();

参数说明

无。

3.16.20 BuildAiCpuOp 函数

函数功能

构建AI CPU的Op。

函数原型

Status BuildAiCpuOp();

参数说明

无。

3.16.21 AddOutputDescs 函数

函数功能

将GetOutputDesc获取到的算子输出描述添加OpDef中。

函数原型

inline Status AddOutputDescs(vector<TensorDescriptor>& v_output_desc);

参数说明

参数	输入/输出	说明
v_output_desc	输入	输出tensor的描述

3.16.22 HasQuantizeDescriptor 函数

函数功能

是否有需要量化的算子，如果有返回true，否则返回false。

函数原型

bool HasQuantizeDescriptor();

参数说明

无。

3.16.23 IsOutputTop 函数

函数功能

检查当前layer的top是否为用户指定的网络输出top，若是则返回true，否则返回false。

函数原型

bool IsOutputTop(string op_name,int32_t index);;

参数说明

参数	输入/输出	说明
op_name	输入	算子名称
index	输入	指定的top

3.16.24 SetRealdimcntForType 函数

函数功能

tensorflow设置真实realdimcnt，其他框架默认为4。

函数原型

Status SetRealdimcntForType(vector<TensorDescriptor>& v_output_desc);

参数说明

参数	输入/输出	说明
v_output_desc	输入	输出tensor描述

3.16.25 SetCaffeTERealdimcnt 函数

函数功能

设置Caffe框架下TE算子的realdimcnt。

函数原型

Status SetCaffeTERealdimcnt(vector<TensorDescriptor>& v_output_desc);

参数说明

参数	输入/输出	说明
v_output_desc	输入	输出tensor描述

3.16.26 SetDefaultRealDimCnt 函数

函数功能

设置realdimcnt，这里设置默认值4。

函数原型

Status SetDefaultRealDimCnt();

参数说明

无。

3.16.27 AdjustConstOpOutputSize 函数

函数功能

设置const op的输出大小。

函数原型

Status AdjustConstOpOutputSize();

参数说明

无。

3.16.28 PadWeight 函数

函数功能

对bias进行16个数对齐。

函数原型

static void PadWeight(WeightDef* weight);

参数说明

参数	输入/输出	说明
weight	输出	WeightDef结构

3.17 TVMOpBuilder 类

3.17.1 TVMOpBuilder 函数

函数功能

构造函数。

函数原型

TVMOpBuilder(OpDef *op_def);

参数说明

参数	输入/输出	说明
op_def	输入	对应算子的OpDef

3.17.2 ~TVMOpBuilder 函数

函数功能

析构函数。

函数原型

~TVMOpBuilder();

参数说明

无。

3.17.3 GetWorkSpaceMemory 函数

函数功能

获取算子的所有workspace内存大小。

函数原型

Status GetWorkSpaceMemory(vector<int64_t>& v_workspace_memory) override;

参数说明

参数	输入/输出	说明
v_workspace_memory	输出	获取的workspace内存大小列表

3.17.4 BuildTvmOp 函数

函数功能

构造tvm 算子。

函数原型

Status BuildTvmOp() override;

参数说明

无。

3.17.5 PackageTvmBinFile 函数

函数功能

从handle中获取json文件名，解析其中数据到当前op中。

函数原型

Status PackageTvmBinFile(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	数据信息

3.17.6 PackageTvmJsonInfo 函数

函数功能

从info中获取json文件路径，并解析其中的数据，并保存到op中。

函数原型

Status PackageTvmJsonInfo(const TVMBinInfo& info);

参数说明

参数	输入/输出	说明
info	输入	TVMBinInfo格式数据

3.17.7 ReadJsonFile 函数

函数功能

从file路径中读取json数据到handle中。

函数原型

Status ReadJsonFile(const std::string& file, JsonHandle handle);

参数说明

参数	输入/输出	说明
file	输入	文件路径名
handle	输出	解析出来的json数据

3.17.8 ParseTvmMagic 函数

函数功能

解析handle数据，并保存到当前op中。

函数原型

Status ParseTvmMagic(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	json数据信息

3.17.9 ParseTvmBlockDim 函数

函数功能

解析handle数据，并保存到当前op中。

函数原型

Status ParseTvmBlockDim(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	json数据信息

3.17.10 ParseTvmWorkSpace 函数

函数功能

解析workspace信息。

函数原型

Status ParseTvmWorkSpace(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	json数据信息

3.17.11 ParseTvmMetaData 函数

函数功能

解析handle数据，并保存到当前op中。

函数原型

Status ParseTvmMetaData(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	json数据信息

3.17.12 ParseTvmKernelName 函数

函数功能

解析handle数据，并保存到当前op中。

函数原型

Status ParseTvmKernelName(const JsonHandle handle);

参数说明

参数	输入/输出	说明
handle	输入	json数据信息

3.17.13 GetStrValueFromJson 函数

函数功能

从handle中获取key对应的json数据，并返回。

函数原型

std::string GetStrValueFromJson(const JsonHandle handle, const std::string& key);

参数说明

参数	输入/输出	说明
handle	输入	json数据
key	输入	键值

3.18 ModelParser 类

3.18.1 ModelParser 函数

函数功能

构造函数。

函数原型

explicit ModelParser();

参数说明

无。

3.18.2 ~ModelParser 函数

函数功能

析构函数。

函数原型

virtual ~ModelParser();

参数说明

无。

3.18.3 Parse 函数

函数功能

纯虚函数，需要子类实现；解析网络模型数据。

函数原型

virtual Status Parse(const char* file, Graph *graph) = 0;

参数说明

参数	输入/输出	说明
file	输入	网络模型文件路径
graph	输出	保存解析后网络信息的图

3.18.4 ParseProto 函数

函数功能

纯虚函数，需要子类实现；用于解析网络模型数据。

函数原型

virtual Status ParseProto(google::protobuf::Message* proto, Graph *graph) = 0;

参数说明

参数	输入/输出	说明
proto	输入	Proto格式的网络模型
graph	输出	保存解析后网络信息的图

3.18.5 ToJson 函数

函数功能

将模型文件转换为JSON格式。

函数原型

virtual Status ToJson(const char* model_file, const char* json_file);

参数说明

参数	输入/输出	说明
model_file	输入	待转换的模型文件路径
json_file	输出	转换后的JSON文件路径

3.19 OpParser 类

3.19.1 ~OpParser 函数

函数功能

析构函数。

函数原型

virtual ~OpParser();

参数说明

无。

3.19.2 ParseParams 函数

函数功能

解析算子参数。

函数原型

virtual Status ParseParams(const Message* op_src, OpDef* op_dest) = 0;

参数说明

参数	输入/输出	说明
op_src	输入	待解析的参数数据
op_dest	输出	解析后的参数数据

3.19.3 ParseWeights 函数

函数功能

解析算子权重信息。

函数原型

virtual Status ParseWeights(const Message* op_src, OpDef* op_dest) = 0;

参数说明

参数	输入/输出	说明
op_src	输入	待解析的权重数据
op_dest	输出	解析后的权重数据

3.19.4 GetFormat 函数

函数功能

根据算子中的参数，获取format信息。

函数原型

virtual Status GetFormat(const Message* op_src, domiTensorFormat_t& format);

参数说明

参数	输入/输出	说明
op_src	输入	待解析的参数数据
format	输出	解析出的format信息

3.20 OpParserFactory 类

3.20.1 Instance 函数

函数功能

根据框架类型获取Parser工厂实例。

函数原型

static shared_ptr<OpParserFactory> Instance(FrameworkType framework);

参数说明

参数	输入/输出	说明
framework	输入	框架类型

3.20.2 CreateOpParser 函数

函数功能

根据输入的type，创建OpParser。

函数原型

shared_ptr<OpParser> CreateOpParser(const std::string& op_type);

参数说明

参数	输入/输出	说明
op_type	输入	算子类型

3.20.3 GetSameTypeOps 函数

函数功能

获取相同类型的OP map，并返回。

函数原型

const map<std::string, std::vector<std::string>>& GetSameTypeOps();

参数说明

无。

3.20.4 ~OpParserFactory 函数

函数功能

析构函数。

函数原型

~OpParserFactory();

参数说明

无。

3.20.5 OpParserFactory 函数

函数功能

构造函数。

函数原型

OpParserFactory();

参数说明

无。

3.20.6 RegisterCreator 函数

函数功能

注册parser函数。

函数原型

void RegisterCreator(const std::string& type, CREATOR_FUN fun);

参数说明

参数	输入/输出	说明
type	输入	算子类型
fun	输入	OpParser的创建函数

3.20.7 RegisterSameTypeOp 函数

函数功能

注册相同类型的算子。

函数原型

void RegisterSameTypeOp(const std::string& same_op_type, const std::string& child_op_type);

参数说明

参数	输入/输出	说明
same_op_type	输入	算子类型
child_op_type	输入	候选算子类型

3.21 OpParserRegisterar 类

3.21.1 OpParserRegisterar 函数

函数功能

构造函数，注册对应框架的parser工厂。

函数原型

OpParserRegisterar(FrameworkType framework, const std::string& op_type, OpParserFactory::CREATOR_FUN fun);

参数说明

参数	输入/输出	说明
framework	输入	框架类型
op_type	输入	算子类型
fun	输入	Parser工厂creator函数

3.21.2 ~OpParserRegisterar 函数

函数功能

析构函数。

函数原型

~OpParserRegisterar();

参数说明

无。

3.22 SameTypeOPRegisterar 类

3.22.1 SameTypeOPRegisterar 函数

函数功能

构造函数；为相同类型的Op注册Map。

函数原型

SameTypeOPRegisterar(FrameworkType framework, const std::string& same_op_type, const std::string& child_op_type);

参数说明

参数	输入/输出	说明
framework	输入	框架类型
same_op_type	输入	算子类型
child_op_type	输入	候选算子类型

3.22.2 ~SameTypeOPRegisterar 函数

函数功能

析构函数。

函数原型

~SameTypeOPRegisterar();

参数说明

无。

3.23 WeightsParser 类

3.23.1 WeightsParser 函数

函数功能

构造函数。

函数原型

WeightsParser();

参数说明

无。

3.23.2 ~WeightsParser 函数

函数功能

析构函数。

函数原型

virtual ~WeightsParser();

参数说明

无。

3.23.3 Parse 函数

函数功能

解析权重数据。

函数原型

virtual Status Parse(const char* file, Graph* graph) = 0;

参数说明

参数	输入/输出	说明
file	输入	训练后权重文件的路径
graph	输出	保存解析后权重信息的图

3.24 CaffeOpParser 类

3.24.1 ParseParams 函数

函数功能

解析算子参数信息。

函数原型

Status ParseParams(const Message* op_src, OpDef* op_dest);

参数说明

参数	输入/输出	说明
op_src	输入	待解析的proto数据
op_dest	输出	解析后的op信息

3.24.2 ParseWeights 函数

函数功能

解析算子权重信息。

函数原型

Status ParseWeights(const Message* op_src, OpDef* op_dest);

参数说明

参数	输入/输出	说明
op_src	输入	待解析的权重数据
op_dest	输出	解析后的权重数据

3.24.3 ConvertWeight 函数

函数功能

将BlobProto转换为WeightDef。

函数原型

static Status ConvertWeight(const BlobProto& proto, WeightDef* weight);

参数说明

参数	输入/输出	说明
proto	输入	待转换的Caffe权重信息
weight	输出	转换后的权重信息

3.24.4 ConvertShape 函数

函数功能

将BlobProto转换为ShapeDef。

函数原型

static void ConvertShape(const BlobProto& proto, ShapeDef* shape);

参数说明

参数	输入/输出	说明
proto	输入	转换前的Shape信息
shape	输出	保存转换的Shape信息

3.25 CaffeWeightParserBuilder 类

3.25.1 ~CaffeWeightParserBuilder 函数

函数功能

析构函数。

函数原型

virtual ~CaffeWeightParserBuilder();

参数说明

无。

3.25.2 AddWeightParserConfig 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual CaffeWeightParserBuilder& AddWeightParserConfig() = 0;

参数说明

无。

3.25.3 CaffeWeightParserBuilder 函数

函数功能

纯虚函数，需要子类实现

函数原型

virtual CaffeWeightParserBuilder& Required() = 0;

参数说明

无。

3.25.4 Optional 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual CaffeWeightParserBuilder& Optional() = 0;

参数说明

无。

3.25.5 OriginShapeSize 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual CaffeWeightParserBuilder& OriginShapeSize(uint32_t origin_shape_size) = 0;

参数说明

参数	输入/输出	说明
origin_shape_size	输入	shape的size，单位是字节

3.25.6 ShapeUseOrigin 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual CaffeWeightParserBuilder& ShapeUseOrigin() = 0;

参数说明

无。

3.25.7 SetShapeMapFn 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual CaffeWeightParserBuilder& SetShapeMapFn(ShapeMapFn shape_map_fn) = 0;

参数说明

参数	输入/输出	说明
shape_map_fn	输入	shape map 函数指针

3.26 CaffeParserBuilder 类

3.26.1 CaffeParserBuilder 函数

函数功能

构造函数。

函数原型

CaffeParserBuilder(const std::string& om_optype);

参数说明

参数	输入/输出	说明
om_optype	输入	算子类型

3.26.2 ~CaffeParserBuilder 函数

函数功能

析构函数。

函数原型

virtual ~CaffeParserBuilder();

参数说明

无

3.26.3 SetParseParamsFn 函数

函数功能

设置参数解析函数。

函数原型

CaffeParserBuilder& SetParseParamsFn(ParseParamsFn parse_params_fn);

参数说明

参数	输入/输出	说明
parse_params_fn	输入	参数解析函数指针

3.26.4 AddWeightParserConfig 函数

函数功能

添加权重解析配置。

函数原型

CaffeWeightParserBuilder& AddWeightParserConfig();

参数说明

无。

3.26.5 Required 函数

函数功能

设置required字段为true。

函数原型

virtual CaffeWeightParserBuilder& Required() override;

参数说明

无。

3.26.6 Optional 函数

函数功能

设置required字段为false。

函数原型

virtual CaffeWeightParserBuilder& Optional() override;

参数说明

无。

3.26.7 OriginShapeSize 函数

函数功能

设置shape size。

函数原型

virtual CaffeWeightParserBuilder& OriginShapeSize(uint32_t origin_shape_size) override;

参数说明

参数	输入/输出	说明
origin_shape_size	输入	shape size

3.26.8 ShapeUseOrigin 函数

函数功能

设置use origin为true。

函数原型

virtual CaffeWeightParserBuilder& ShapeUseOrigin() override;

参数说明

无。

3.26.9 SetShapeMapFn 函数

函数功能

设置shape map函数指针。

函数原型

virtual CaffeWeightParserBuilder& SetShapeMapFn(ShapeMapFn shape_map_fn) override;

参数说明

参数	输入/输出	说明
shape_map_fn	输入	shape map函数指针

3.26.10 SetParseWeightsFn 函数

函数功能

设置权重解析函数。

函数原型

CaffeParserBuilder& SetParseWeightsFn(ParseWeightsFn parser_weights_fn);

参数说明

参数	输入/输出	说明
parser_weights_fn	输入	权重解析函数指针

3.26.11 Finalize 函数

函数功能

注册一个算子的parser。

函数原型

virtual bool Finalize() override;

参数说明

无。

3.27 CaffeOpParserAdapter 类

3.27.1 CaffeOpParserAdapter 函数

函数功能

构造函数。

函数原型

CaffeOpParserAdapter(CaffeParserBuilder<Param> builder);

参数说明

参数	输入/输出	说明
Builder	输入	Parser build信息

3.27.2 ~CaffeOpParserAdapter 函数

函数功能

析构函数。

函数原型

virtual ~CaffeOpParserAdapter();

参数说明

无

3.27.3 ParseParams 函数

函数功能

算子参数解析函数。

函数原型

virtual Status ParseParams(const Message* op_src, OpDef* op_dest) override;

参数说明

参数	输入/输出	说明
op_src	输入	待解析的参数数据
op_dest	输出	解析后的算子参数数据

3.27.4 ParseWeights 函数

函数功能

算子权重解析函数。

函数原型

virtual Status ParseWeights(const Message* op_src, OpDef* op_dest) override;

参数说明

参数	输入/输出	说明
op_src	输入	待解析的权重数据
op_dest	输出	解析后的权重数据

3.28 Finalizeable 类

3.28.1 Finalize 函数

函数功能

纯虚函数，需要子类实现。

函数原型

virtual bool Finalize() = 0;

参数说明

无。

3.28.2 ~Finalizeable 函数

函数功能

析构函数。

函数原型

virtual ~Finalizeable();

参数说明

无。

3.29 Receiver 类

3.29.1 Receiver 函数

函数功能

构造函数。

函数原型

Receiver(Finalizeable& f);

参数说明

参数	输入/输出	说明
f	输入	Finalizeable对象

3.29.2 ~Receiver 函数

函数功能

析构函数。

函数原型

~Receiver();

参数说明

无。

3.30 TensorFlowFusionOpParser 类

3.30.1 ParseParams 函数

函数功能

解析算子参数。

函数原型

virtual Status ParseParams(const vector<const NodeDef*>& v_input_const, OpDef* op_dest);

参数说明

参数	输入/输出	说明
v_input_const	输入	待解析的算子参数
op_dest	输出	解析后的模型数据

3.30.2 ParseParams 函数（带 final）

函数功能

解析算子参数。

函数原型

Status ParseParams(const Message* op_src, OpDef* op_dest) final;

参数说明

参数	输入/输出	说明
v_input_const	输入	待解析的算子参数
op_dest	输出	解析后的模型数据

3.30.3 ParseWeights 函数

函数功能

解析权值信息。

函数原型

Status ParseWeights(const vector<const NodeDef*>& v_input_const, OpDef* op_dest);

参数说明

参数	输入/输出	说明
v_input_const	输入	待解析的权值数据
op_dest	输出	解析后权值数据

3.30.4 ParseParamFromConst 函数

函数功能

从const op中解析参数。

函数原型

Status ParseParamFromConst(const NodeDef* input_const , int32_t & param);

Status ParseParamFromConst(const NodeDef* nodeDef , int32_t & param ,int index);

Status ParseParamFromConst(const NodeDef* input_const , float & param);

Status ParseParamFromConst(const NodeDef* nodeDef , float & param ,int index);

参数说明

参数	输入/输出	说明
input_const或nodeDef	输入	节点信息
param	输出	解析出的参数信息
Index	输入	下标索引

3.30.5 GetTensorFromNode 函数

函数功能

从node中获取tensor信息。

函数原型

Status GetTensorFromNode(const NodeDef* nodeDef , TensorProto& tensor);

参数说明

参数	输入/输出	说明
nodeDef	输入	node信息
tensor	输出	获得的tensor信息

3.31 TensorFlowOpParser 类

3.31.1 ParseWeights 函数（2 个参数）

函数功能

解析节点中的权重参数。

函数原型

virtual Status ParseWeights(const vector<const NodeDef*>& v_input_const, OpDef* op_dest);

Status ParseWeights(const Message* op_src, OpDef* op_dest) final;

参数说明

参数	输入/输出	说明
v_input_const/ op_src	输入	待解析的节点数据
op_dest	输出	保存解析后节点数据

3.31.2 ParseWeights 函数（3 个参数）

函数功能

解析节点中的权重参数。

函数原型

virtual Status ParseWeights(const vector<const NodeDef*>& input_nodes, OpDef* op_dest, vector<OpDef*>& input_ops);

参数说明

参数	输入/输出	说明
input_nodes	输入	待解析的节点数据
op_dest	输出	保存解析后节点数据
input_ops	输入	输入Op信息

3.31.3 ParseParams 函数

函数功能

解析算子参数。

函数原型

Status ParseParams(const Message* op_src, OpDef* op_dest);

参数说明

参数	输入/输出	说明
op_src	输入	待解析的算子数据
op_dest	输出	解析后的算子数据

3.31.4 ParseDataType 函数（3 个参数）

函数功能

解析数据类型。

函数原型

Status ParseDataType(const NodeDef *node_src, const string &attr_src, DataType &data_type);

参数说明

参数	输入/输出	说明
node_src	输入	待解析的Node
attr_src	输入	待解析的Attribute
data_type	输出	解析后的数据类型

3.31.5 ParseDataType 函数（4 个参数）

函数功能

解析数据类型。

函数原型

Status ParseDataType(const NodeDef *node_src, const string &attr_src,OpDef *op_dest, const string &attr_dest);

参数说明

参数	输入/输出	说明
node_src	输入	待解析的Node
attr_src	输入	待解析的Attribute
op_dest	输出	解析后的Op
attr_dest	输出	解析后的attr_dest

3.31.6 ConvertWeight 函数

函数功能

将NodeDef转换为WeightDef。

函数原型

Status ConvertWeight(const NodeDef* nodeDef, WeightDef* weight);

参数说明

参数	输入/输出	说明
nodeDef	输入	待转换的TensorFlow权重信息
weight	输出	转换后的权重信息

3.31.7 SetWeightData 函数

函数功能

将NodeDef中的tensor值取出放入weightdef。

函数原型

Status SetWeightData(const TensorProto& tensor, WeightDef* weight);

参数说明

参数	输入/输出	说明
tensor	输入	Tensor信息
weight	输出	转换后的权重信息

3.31.8 TensorflowDataTypeToDomi 函数

函数功能

将data type转为domi域下的data type，并返回。

函数原型

static domiDataType_t TensorflowDataTypeToDomi(DataType tf_type);

参数说明

参数	输入/输出	说明
tf_type	输入	原始data type

3.32 TensorFlowUtil 类

3.32.1 FindAttrValue 函数

函数功能

在NodeDef中查找对应的AttrValue，若找到则返回true，否则返回false。

函数原型

static bool FindAttrValue(const tensorflow::NodeDef* nodeDef, const string attr_name, tensorflow::AttrValue& attr_value);

参数说明

参数	输入/输出	说明
nodeDef	输入	需要查找的Nodedef对象
attr_name	输入	属性name
attr_value	输出	属性Value对象

3.32.2 CheckAttrHasType 函数

函数功能

检查AttrValue属性的实际类型和期望类型，int、float、list(int)、list(bool)等。

函数原型

static Status CheckAttrHasType(const AttrValue& attr_value, string type);

参数说明

参数	输入/输出	说明
attr_value	输入	属性值
type	输入	期望的type

3.32.3 ParseDataType 函数

函数功能

解析数据类型。

函数原型

static Status ParseDataType(const NodeDef *node_src, const string &attr_src, DataType &data_type);

参数说明

参数	输入/输出	说明
node_src	输入	待解析的Node
attr_src	输入	待解析的Attribute
data_type	输出	解析后的数据类型

3.32.4 TransTensorDescriptor 函数

函数功能

将NodeDef中attr 里面的一个attr_value 存储到OpDef中。

函数原型

static Status TransTensorDescriptor(const AttrValue attr_value, Operator *op, const uint32_t io, string type="");

参数说明

参数	输入/输出	说明
attr_value	输入	待转换的NodeDef中Attr
op	输出	解析后的信息存储在父类的属性中
io	输入	表示是inputtensor/ outputtensor
type	输入	算子类型

3.32.5 AddNodeAttr 函数

函数功能

添加NodeDef属性。

函数原型

static void AddNodeAttr(const string& attr_name, const tensorflow::AttrValue& value, tensorflow::NodeDef* node_def);

参数说明

参数	输入/输出	说明
attr_name	输入	属性名称
value	输入	该属性的值
node_def	输出	将属性添加到这个NodeDef

3.32.6 ClearUnusedParam 函数

函数功能

清除graph中无用参数。

函数原型

static Status ClearUnusedParam(Graph* graph);

参数说明

参数	输入/输出	说明
graph	输入	Graph的对象

3.33 GraphToFunctionDef 类

3.33.1 RecordArg 函数

函数功能

为functiondef添加输入算子。

函数原型

static Status RecordArg(Graph* graph ,vector<Edge *> input_edges);

参数说明

参数	输入/输出	说明
graph	输出	FunctionDef对应的子图
input_edges	输入	FunctionDef对应的输入边

3.33.2 RecordResult 函数

函数功能

用为functiondef添加输出算子。

函数原型

static Status RecordResult(Graph* graph ,vector<Edge *> output_edges);

参数说明

参数	输入/输出	说明
graph	输出	FunctionDef对应的子图
output_edges	输入	FunctionDef对应的输出边

3.33.3 DavGraphToFunctionDef 函数

函数功能

Graph转为FunctionDef。

函数原型

static Status DavGraphToFunctionDef(Graph* graph, const string& name, FunctionDef* fdef);

参数说明

参数	输入/输出	说明
graph	输入	FunctionDef对应的子图
name	输入	FunctionDef名称
fdef	输出	转换后的FunctionDef

3.33.4 BuildFunctionDef 函数

函数功能

创建FunctionDef。

函数原型

static Status BuildFunctionDef(Graph* graph,

const string name_in,

FunctionDefLibrary* library,

NodeDef* call_node_def,

vector<Edge *> input_edges,

vector<Edge *> output_edges);

参数说明

参数	输入/输出	说明
graph	输入	FunctionDef对应的子图
name_in	输入	FunctionDef名称
library	输出	创建的FunctionDef
call_node_def	输出	创建的functiondef对应的node def
input_edges	输入	FunctionDef对应的输入边
output_edges	输入	FunctionDef对应的输出边

3.34 NodeNameMapping 类

3.34.1 NodeNameMapping 函数

函数功能

构造函数。

函数原型

NodeNameMapping() = default;

参数说明

无。

3.34.2 Normalize 函数

函数功能

将输入/输出node名称统一，并使其唯一。

函数原型

string Normalize(const string& name);

参数说明

参数	输入/输出	说明
name	输入	node名称

3.34.3 Uniquify 函数

函数功能

使name唯一。

函数原型

string Uniquify(const string& name);

参数说明

参数	输入/输出	说明
name	输入	名称

3.34.4 Renormalize 函数

函数功能

返回name的rename名称，如果不存在则返回空。

函数原型

string Renormalize(const string& name) const;

参数说明

参数	输入/输出	说明
name	输入	查询的名称

3.34.5 NormalizeHelper 函数

函数功能

将字符串中的字母都转为小写，非字母转为_。

函数原型

string NormalizeHelper(string name) const;

参数说明

参数	输入/输出	说明
name	输入	输入字符串

3.34.6 UniquifyHelper 函数

函数功能

构造一个唯一的字符串。

函数原型

string UniquifyHelper(string name);

参数说明

参数	输入/输出	说明
name	输入	输入字符串

3.35 Tuple 类

3.35.1 Tuple 函数

函数功能

构造函数。

函数原型

Tuple() = default;

参数说明

无。

3.35.2 ~Tuple 函数

函数功能

析构函数。

函数原型

inline ~Tuple();

参数说明

无。

3.35.3 Tuple 复制构造函数

函数功能

复制构造函数。

函数原型

inline Tuple(const Tuple< ValueType >& s);

参数说明

参数	输入/输出	说明
s	输入	另一个tuple对象

3.35.4 Tuple 函数（1 个入参）

函数功能

构造函数。

函数原型

inline Tuple(std::initializer_list< ValueType > init);

inline Tuple(std::vector< ValueType > init);

inline Tuple(Tuple< ValueType >&& src);

参数说明

参数	输入/输出	说明
init	输入	● std::initializer_list< ValueType > init：从一个list构造tuple ● std::vector< ValueType > init：从一个vector结构构造tuple
src	输入	另一个tuple对象

3.35.5 Tuple 函数（2 个入参）

函数功能

从一个迭代器构造Tuple。

函数原型

inline Tuple(RandomAccessIterator begin, RandomAccessIterator end);

参数说明

参数	输入/输出	说明
begin	输入	迭代器开始
end	输入	迭代器结束

3.35.6 assign 函数

函数功能

赋值一个迭代器内容到Tuple。

函数原型

inline void assign(RandomAccessIterator begin, RandomAccessIterator end);

参数说明

参数	输入/输出	说明
begin	输入	迭代器开始
end	输入	迭代器结束

3.35.7 swap 函数

函数功能

交换数据。

函数原型

inline void swap(Tuple< ValueType >& other);

参数说明

参数	输入/输出	说明
other	输入	另一个Tuple对象

3.35.8 operator=函数

函数功能

赋值构造函数。

函数原型

inline Tuple< ValueType >& operator=(const Tuple< ValueType >& src);

inline Tuple< ValueType >& operator=(Tuple< ValueType >&& src);

inline Tuple< ValueType >& operator=(std::initializer_list< ValueType > init);

参数说明

参数	输入/输出	说明
src	输入	另一个Tuple对象
init	输入	list对象

3.35.9 operator==函数

函数功能

判断两个Tuple对象是否相等。

函数原型

inline bool operator==(const Tuple< ValueType >& s) const;

参数说明

参数	输入/输出	说明
s	输入	另一个Tuple对象

3.35.10 operator!=函数

函数功能

判断两个Tuple对象是否相等。

函数原型

inline bool operator!=(const Tuple< ValueType >& s) const;

参数说明

参数	输入/输出	说明
s	输入	另一个Tuple对象

3.35.11 begin 函数

函数功能

返回数据的头指针。

函数原型

inline const ValueType* begin() const;

参数说明

无。

3.35.12 begin 函数（非 const）

函数功能

返回数据的头指针。

函数原型

inline ValueType* begin();

参数说明

无。

3.35.13 end 函数

函数功能

返回数据的尾指针。

函数原型

inline const ValueType* end() const;

参数说明

无。

3.35.14 end 函数（非 const）

函数功能

返回数据的尾指针。

函数原型

inline ValueType* end();

参数说明

无。

3.35.15 ndim 函数

函数功能

返回Tuple的维度。

函数原型

inline uint32_t ndim() const;

参数说明

无。

3.35.16 operator[]函数

函数功能

数组索引重写，返回第i个tuple值。

函数原型

inline ValueType& operator[](size_t i);

参数说明

参数	输入/输出	说明
i	输入	索引

3.35.17 operateor[]函数（const）

函数功能

数组索引重写，返回第i个tuple值。

函数原型

inline const ValueType& operator[](size_t i) const;

参数说明

参数	输入/输出	说明
i	输入	索引

3.35.18 operator<<函数

函数功能

重写<<运算符，输出Tuple对象。

函数原型

friend std::ostream& operator<<(std::ostream& os, const Tuple< ValueType >& t);

参数说明

参数	输入/输出	说明
os	输入	流
t	输入	被输出对象

3.35.19 operator>>函数

函数功能

重写>>运算符，输入Tuple对象。

函数原型

friend std::istream& operator>>(std::istream& is, Tuple< ValueType >& t);

参数说明

参数	输入/输出	说明
is	输入	流
t	输入	被输入对象

3.35.20 SetDim 函数

函数功能

设置dim。

函数原型

inline void SetDim(uint32_t ndim);

参数说明

参数	输入/输出	说明
ndim	输入	需要设置的dim

3.36 CaffeTVMOpParser 类

3.36.1 ParseWeights 函数

函数功能

权重解析函数。

函数原型

virtual Status ParseWeights(const Message* op_src, OpDef* op_dest) override;

参数说明

参数	输入/输出	说明
op_src	输入	原始输入算子信息
op_dest	输出	目标算子信息

4 宏定义

4.1 错误码生成宏

宏功能

调用DEF_ERRORNO生成指定模块的错误码。

宏原型

DEF_ERRORNO(sysid, modid, name, value, desc)

参数说明

参数	输入/输出	说明
sysid	输入	系统id
modid	输入	模块id
name	输入	错误码名称
value	输入	错误码的实际值
desc	输入	错误码描述字符串

4.2 COMMON 模块错误码生成宏

宏功能

调用DEF_ERRORNO_COMMON生成common模块的错误码。

宏原型

DEF_ERRORNO_COMMON(name, value, desc)

参数说明

参数	输入/输出	说明
name	输入	错误码名称
value	输入	错误码的实际值
desc	输入	错误码描述字符串

4.3 OMG 模块错误码生成宏

宏功能

调用DEF_ERRORNO_OMG生成OMG模块的错误码。

宏原型

DEF_ERRORNO_OMG(name, value,desc)

参数说明

参数	输入/输出	说明
name	输入	错误码名称
value	输入	错误码的实际值
desc	输入	错误码描述字符串

4.4 OME 模块错误码生成宏

宏功能

调用DEF_ERRORNO_OME生成OME模块的错误码。

宏原型

DEF_ERRORNO_OME(name, value,desc)

参数说明

参数	输入/输出	说明
name	输入	错误码名称
value	输入	错误码的实际值
desc	输入	错误码描述字符串

4.5 CALIBRATION 模块错误码生成宏

宏功能

调用DEF_ERRORNO_CALIBRATION生成量化模块的错误码。

宏原型

DEF_ERRORNO_CALIBRATION(name, value, desc)

参数说明

参数	输入/输出	说明
name	输入	错误码名称
value	输入	错误码的实际值
desc	输入	错误码描述字符串

4.6 获取错误码描述宏

宏功能

调用GET_ERRORNO_STR获取错误码描述。

宏原型

GET_ERRORNO_STR(value)

参数说明

参数	输入/输出	说明
value	输入	错误码的值

4.7 算子类型注册宏

宏功能

对一种算子类型进行注册，主要是声明一个常量字符串,该字符串就是算子类型的名称，然后将其保存到一个set中。

宏原型

REGISTER_OPTYPE(var_name, str_name)

参数说明

参数	输入/输出	说明
var_name	输入	常量字符串名称
str_name	输入	算子类型名称

4.8 算子模式类型注册宏

宏功能

注册算子模型，即定义一个常量字符串。

宏原型

REGISTER_OPSAMETYPE(var_name, str_name)

参数说明

参数	输入/输出	说明
var_name	输入	常量字符串名称
str_name	输入	该算子模型的名称

4.9 算子类型是否存在宏

宏功能

检查一个算子类型是否存在。

宏原型

IS_OPTYPE_EXISTING(str_name)

参数说明

参数	输入/输出	说明
str_name	输入	算子类型名称

4.10 算子模式注册宏

4.10.1 DOMI_OP_SCHEMA 宏

宏功能

注册一种叫name的OpSchema。

宏原型

DOMI_OP_SCHEMA(name)

参数说明

参数	输入/输出	说明
name	输入	算子模式名称

4.10.2 DOMI_OP_SCHEMA_UNIQ_HELPER 宏

宏功能

被DOMI_OP_SCHEMA调用。

宏原型

DOMI_OP_SCHEMA_UNIQ_HELPER(ctr, name)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子模式名称

4.10.3 DOMI_OP_SCHEMA_UNIQ 宏

宏功能

被DOMI_OP_SCHEMA_UNIQ_HELPER调用。

宏原型

DOMI_OP_SCHEMA_UNIQ(ctr, name)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子模式名称

4.11 算子注册宏

4.11.1 DOMI_REGISTER_OP 宏

宏功能

注册一种叫name的算子类型。

宏原型

DOMI_REGISTER_OP(name)

参数说明

参数	输入/输出	说明
name	输入	算子类型名称

4.11.2 DOMI_REGISTER_OP_UNIQ_HELPER 宏

宏功能

被DOMI_REGISTER_OP调用，用于注册算子类型。

宏原型

DOMI_REGISTER_OP_UNIQ_HELPER(ctr, name)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子类型名称

4.11.3 DOMI_REGISTER_OP_UNIQ 宏

宏功能

被DOMI_REGISTER_OP_UNIQ_HELPER调用，用于注册算子类型。

宏原型

DOMI_REGISTER_OP_UNIQ(ctr, name)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子类型名称

4.12 算子 Parser 宏

宏功能

返回算子Parser的对象指针。

宏原型

PARSER_FN(clazz)

参数说明

参数	输入/输出	说明
clazz	输入	算子对应的Parser类的名称

4.13 算子 Builder 宏

宏功能

返回算子Builder的对象指针。

宏原型

BUILDER_FN(clazz)

参数说明

参数	输入/输出	说明
clazz	输入	算子对应的Builder类的名称

4.14 OpParser 的注册宏

宏功能

OpParser的注册宏。

宏原型

REGISTER_OP_PARSER_CREATOR(framework, op_type, clazz)

参数说明

参数	输入/输出	说明
framework	输入	框架类型
op_type	输入	算子类型
clazz	输入	OpParser的实现类

4.15 相同类型的 OpParser 注册宏

宏功能

为相同类型的OpParser注册。

宏原型

REGISTER_OP_PARSER_CREATOR_SAME_TYPE(framework, same_op_type, child_op_type, clazz)

参数说明

参数	输入/输出	说明
framework	输入	框架类型
same_op_type	输入	算子类型
child_op_type	输入	候选算子类型

参数	输入/输出	说明
clazz	输入	OpParser的实现类

4.16 Caffe 算子解析注册宏

4.16.1 DOMI_REGISTER_CAFFE_PARSER 宏

宏功能

将caffe算子的参数、权重解析函数进行注册。

宏原型

DOMI_REGISTER_CAFFE_PARSER(name, param_clazz)

参数说明

参数	输入/输出	说明
name	输入	算子类型
param_clazz	输入	对应算子解析的类

4.16.2 DOMI_REGISTER_CAFFE_PARSER_UNIQ_HELPER 宏

宏功能

被DOMI_REGISTER_CAFFE_PARSER调用，注册caffe算子。

宏原型

DOMI_REGISTER_CAFFE_PARSER_UNIQ_HELPER(ctr, name, param_clazz)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子类型
param_clazz	输入	对应算子解析的类

4.16.3 DOMI_REGISTER_CAFFE_PARSER_UNIQ 宏

宏功能

被DOMI_REGISTER_CAFFE_PARSER_UNIQ_HELPER调用，注册caffe算子。

宏原型

DOMI_REGISTER_CAFFE_PARSER_UNIQ(ctr, name, param_clazz)

参数说明

参数	输入/输出	说明
ctr	输入	计数值
name	输入	算子类型
param_clazz	输入	对应算子解析的类

4.17 设置属性函数声明宏

4.17.1 ATTR_SETTER_WITH_VALUE 宏

宏功能

Operator中声明不同类型Attr函数。

宏原型

ATTR_SETTER_WITH_VALUE(TypeName)

参数说明

参数	输入/输出	说明
TypeName	输入	数据类型

4.17.2 ATTR_SETTER_WITH_DEFAULT_VALUE 宏

宏功能

OpSchema中声明不同类型Attr函数。

宏原型

ATTR_SETTER_WITH_DEFAULT_VALUE(TypeName)

参数说明

参数	输入/输出	说明
TypeName	输入	数据类型

5 离线模型数据类型介绍

以Mind Studio安装用户登录Mind Studio所在的服务器，可以“tools/che/ddk/ddk/include/inc/custom/proto/om.proto”文件中查看枚举值类型和Message类型。

5.1 Enum 类型

5.1.1 TargetType

功能

标识平台类型的枚举。

成员介绍

成员	类型	说明	枚举值
MINI	枚举	Mini平台	0
TINY	枚举	Tiny平台（预留）	1
LITE	枚举	Lite平台（预留）	2

5.1.2 QuantizeScaleType

功能

标识权值量化后scale的类型。

成员介绍

成员	类型	说明	枚举值
VECTOR_SCALE	枚举	向量	0
SCALAR_SCALE	枚举	标量	1

5.1.3 QuantizeScaleMode

功能

标识权值量化后scale的模式。

成员介绍

成员	类型	说明	枚举值
NORMAL_MODE	枚举	scale为原值	0
SQRT_MODE	枚举	scale为开根号后的值	1

5.1.4 QuantizeAlgorithm

功能

量化算法。

成员介绍

成员	类型	说明	枚举值
NON_OFFSET_ALGO	枚举	数据和权值都不带offset	0
HALF_OFFSET_ALGO	枚举	数据带offset，权值不带offset	1
ALL_OFFSET_ALGO	枚举	数据和权值都带offset	2

5.1.5 DeviceType

功能

设备类型。

成员介绍

成员	类型	说明	枚举值
NPU	枚举	NPU	0
CPU	枚举	CPU	1

5.2 Message 类型

5.2.1 ModelDef

功能

离线模型的message定义。

成员介绍

成员	类型	说明
name	string	模型名称
version	uint32	模型版本号
memory_size	uint64	模型占用的空间
stream_num	uint32	使用的stream个数
event_num	uint32	Event个数
weight_size	uint64	权值占用的空间大小
op	OpDef	算子列表
target_type	TargetType	目标平台类型
attr	map<string, AttrDef>	属性map

5.2.2 OpDef

功能

算子的定义。

成员介绍

成员	类型	说明
name	string	算子名称
type	string	算子类型
id	uint32	编号
stream_id	uint32	所在stream
input_name	string	输入对应的原始op名称+出边索引。op_name：index
src_name	string	输入对应的原始op名称
src_index	int32	输入对应的原始op输出索引
input	int64	输入数据在整体内存里的偏移
output	int64	输出数据在整体内存里的偏移
input_desc	TensorDescriptor	输入数据的描述
output_desc	TensorDescriptor	输出数据的描述
weights	WeightDef	权值数据
dst_name	string	输出对应的原始名称
dst_index	int32	输出对应的原始op输出索引
workspace	int64	Workspace在整体内存里的偏移
workspace_bytes	uint32	Workspace内存的大小
weight_name	string	Weight的名称
is_input_const	bool	input是常量还是变量
attr	map<string, AttrDef>	算子包含的属性map
quantize_factor	QuantizeFactorParams	量化因子

成员	类型	说明
op_params	所选成员的类型	从以下算子参数中选择一个作为成员： ● sender_param：SendOpParams，sender算子参数 ● receiver_param：RecvOpParams，receiver算子参数 ● convolution_param：ConvolutionOpParams，convolution算子参数 ● pooling_param：PoolingOpParams，pooling算子参数 ● eltwise_param：EltwiseOpParams，eltwise算子参数 ● batchnorm_param：BatchNormOpParams，batchnorm算子参数 ● scale_param：ScaleOpParams，scale算子参数 ● full_connection_param：FullConnectionOpParams，full_connection算子参数 ● softmax_param：SoftmaxOpParams，softmax算子参数 ● activation_param：ActivationOpParams，activation算子参数 ● reshape_param：ReshapeOpParams，reshape算子参数

5.2.3 SendOpParams

功能

用于设置Event的message定义。

成员介绍

成员	类型	说明
event_id	uint32	设置的Event

5.2.4 RecvOpParams

功能

用于等待Event的message定义。

成员介绍

成员	类型	说明
event_id	uint32	要同步的Event

5.2.5 QuantizeFactor

功能

量化因子message的定义。

成员介绍

成员	类型	说明
scale_mode	QuantizeScaleMode	scale模式
scale_value	bytes	scale数值
scale_offset	int64	scale在整体内存中的位置
offset_data_value	bytes	数据offset
offset_data_offset	int64	数据offset在整体内存中的位置
offset_weight_value	bytes	权值offset
offset_weight_offset	int64	权值offset在整体内存中的位置
offset_pad_value	bytes	补边值
offset_pad_offset	int64	补边值在整体内存中的位置

5.2.6 QuantizeFactorParams

功能

量化因子参数的message定义。

成员介绍

成员	类型	说明
quantize_algo	QuantizeAlgorithm	量化算法
scale_type	QuantizeScaleType	scale是向量还是标量
quantize_param	QuantizeFactor	量化因子
dequantize_param	QuantizeFactor	反量化因子
requantize_param	QuantizeFactor	重量化因子

5.2.7 ConvolutionOpParams

功能

卷积算子的参数message定义。

成员介绍

成员	类型	说明
mode	int32	卷积模式
algo	int32	卷积前向算法
pad_mode	int32	Padding模式
group	uint32	卷积group
num_output	uint32	算子的输出数量
pad	uint32	Padding值
stride	uint32	卷积步长
dilation	uint32	卷积膨胀值
kernel	uint32	卷积核
alpha	float	比例因子
beta	float	比例因子
filter	WeightDef	Filter数据
bias	WeightDef	Bias偏置数据

成员	类型	说明
relu_flag	bool	relu标志
adj	uint32	用于调整反卷积算子Output的大小，数据按h、w顺序存放
target_shape	uint32	该参数如果设置，则忽略adjs参数，直接使用target_shape的值用作Output的大小
before_pad	uint32	Pad+convolution

5.2.8 PoolingOpParams

功能

池化算子参数的message定义。

成员介绍

成员	类型	说明
mode	int32	池化模式
nan_opt	int32	Nan选项
pad_mode	int32	Padding模式
global_pooling	bool	global_pooling标志
window	uint32	窗大小
pad	uint32	Padding值
stride	uint32	步长
ceil_mode	bool	ceil mode,为true表示向上取整，为false表示向下取整
data_mode	int32	数据类型
alpha	float	比例因子
beta	float	比例因子
before_pad	uint32	Pad+pooling

5.2.9 EltwiseOpParams

功能

Eltwise算子参数的message定义。

成员介绍

成员	类型	说明
mode	int32	element-wise模式
coeff	float	SUM操作的系数
alpha	float	比例因子
beta	float	比例因子
weight	WeightDef	运算常量
relu_flag	bool	relu标志

5.2.10 ActivationOpParams

功能

激活算子参数的message定义。

成员介绍

成员	类型	说明
mode	int32	激活模式
coef	float	参数在chipped RELU中表示上限值，在RELU中表示alpha值，原始RELU中没有使用，填写为默认值0.0
alpha	float	比例因子
beta	float	比例因子

5.2.11 BatchNormOpParams

功能

BatchNorm算子参数的message定义。

成员介绍

成员	类型	说明
mode	int32	BatchNorm模式
alpha	float	比例因子
beta	float	比例因子
epsilon	double	精度值
use_global_stats	bool	全局状态标记，默认true
moving_average_fraction	float	转移平均因子值
estimated_mean	WeightDef	均值估计值
estimated_variance	WeightDef	方差估计值
scale	WeightDef	scale数据
bias	WeightDef	Bias数据

5.2.12 ScaleOpParams

功能

Scale算子参数的message定义。

成员介绍

成员	类型	说明
scale	WeightDef	scale数据
bias	WeightDef	Bias数据

5.2.13 ReshapeOpParams

功能

Reshape算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
shape	ShapeDef	维度数据
axis	int32	需要reshape的维度起始索引
num_axes	int32	需要reshape的维度数
format	int32	算子数据排列方式

5.2.14 SoftmaxOpParams

功能

Softmax算子参数的message定义。

成员介绍

成员	类型	说明
algo	int32	Softmax算法
mode	int32	Softmax模式
alpha	float	比例因子
beta	float	比例因子

5.2.15 FullConnectionOpParams

功能

全连接算子参数的message定义。

成员介绍

成员	类型	说明
filter	WeightDef	Filter数据
bias	WeightDef	Bias数据
num_output	uint32	这一层的输出数量

成员	类型	说明
relu_flag	bool	relu标志

5.2.16 FlattenOpParams

功能

Flatten算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
start_axis	int32	表示从哪个维度开始合并，为从0开始的索引值
end_axis	int32	表示合并结束的维度的索引

5.2.17 AddLimitedOpParams

功能

AddLimited算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
axis	int32	axis索引
broadcast	bool	是否广播
weight	WeightDef	运算常量

5.2.18 MulLimitedOpParams

功能

MulLimited算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
axis	int32	axis索引
broadcast	bool	是否广播
weight	WeightDef	运算常量

5.2.19 AddOpParams

功能

Add算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
weight	WeightDef	运算常量

5.2.20 MulOpParams

功能

Mul算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
weight	WeightDef	运算常量

5.2.21 SubOpParams

功能

Sub算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
weight	WeightDef	运算常量

5.2.22 BiasAddOpParams

功能

BiasAdd算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
bias	WeightDef	运算常量

5.2.23 MatMulOpParams

功能

MatMul算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子
transposeX	bool	是否对X进展转置
transposeW	bool	是否对W进行转置
filter	WeightDef	Filter数据
bias	WeightDef	Bias数据

5.2.24 RsqrtOpParams

功能

Rsqrt算子参数的message定义。

成员介绍

成员	类型	说明
alpha	float	比例因子
beta	float	比例因子

5.2.25 WeightDef

功能

权值描述的message定义。

成员介绍

成员	类型	说明
format	int32	数据格式

成员	类型	说明
data_type	int32	数据类型
shape	ShapeDef	形状/维度
data	bytes	训练后的数据
data_offset	int64	训练数据在整体内存里的偏移
cmps_size	uint32	format为NC1HWC0时为Filter4D转5D重排后的大小，仅对filter压缩模式有效
cmps_tab	bytes	压缩信息表
cmps_tab_offset	int64	压缩信息表在整体内存里的偏移
cmps_info	CompressInfo	压缩分形信息

5.2.26 ShapeDef

功能

形状/维度的描述message定义。

成员介绍

成员	类型	说明
dim	int64	每个维度的大小

5.2.27 TensorDescriptor

功能

Tensor描述符的message定义。

成员介绍

成员	类型	说明
format	int32	Tensor的格式
data_type	int32	Tensor的数据类型
dim	int64	Tensor的每个维度大小

成员	类型	说明
size	uint32	Tensor的数据实际占用空间大小
reuse_input	bool	如果该Tensor是输出，是否直接复用输入的内存
output_tensor	bool	该tensor是否是网络的输出Tensor
device_type	DeviceType	设备类型
input_tensor	bool	该tensor是否是网络的输入Tensor
real_dim_cnt	uint32	实际维度大小
reuse_input_index	uint32	复用内存的输入的序号

5.2.28 CompressInfo

功能

压缩分形信息的message定义。

成员介绍

成员	类型	说明
blockRow	int32	行
blockCol	int32	列
fractalK	int32	K因子
fractalN	int32	N因子
lastFractalK	int32	最后的K因子
lastFractalN	int32	最后的N因子
cubeSize	int32	Cube大小
loadDir	int32	加载方向

5.2.29 AttrDef

功能

属性结构的message定义，包含一种类型的数据。内部定义了ListValue的message定义。

成员介绍

成员	类型	说明
value	所选成员的数据类型	从以下数据类型中选择一个作为成员： ● s：string，字符串类型 ● i：int64，64位整型 ● f：float，浮点型 ● b：bool，布尔型 ● u：uint32，32位整型 ● bt：bytes，字节类型 ● list：ListValue，ListValue类型 ● func：NamedAttrs，NamedAttrs类型

5.2.30 NamedAttrs

功能

属性名称的message定义。

成员介绍

成员	类型	说明
name	string	名字
attr	map<string, AttrDef>	属性map

5.2.31 AippParams

功能

Aipp的message定义。

成员介绍

成员	类型	说明
input_format	InputFormat	输入格式
csc_switch	bool	色域转换开关，静态AIPP配置
load_start_pos_w	int32	抠图起始位置水平方向坐标，抠图大小为网络定义的输入大小

成员	类型	说明
load_start_pos_h	int32	抠图起始位置垂直方向坐标，抠图大小为网络定义的输入大小
src_image_size_w	int32	图像的高度
src_image_size_h	int32	图像的高度
cpadding_value	float	C方向的填充值
rbuv_swap_switch	bool	色域转换前，R通道与B通道交换开关/U通道与V通道交换开关
ax_swap_switch	bool	色域转换前，RGBA->ARGB, YUVA->AYUV交换开关
single_line_mode	bool	单行处理模式（只处理抠图后的第一行）开关
resize	bool	AIPP处理图片时是缩放还是裁剪
mean_chn_0	int32	通道n均值
mean_chn_1	int32	通道n均值
mean_chn_2	int32	通道n均值
min_chn_0	float	通道n最小值
min_chn_1	float	通道n最小值
min_chn_2	float	通道n最小值
var_reci_chn_0	float	通道n方差或(max-min)的倒数
var_reci_chn_1	float	通道n方差或(max-min)的倒数
var_reci_chn_2	float	通道n方差或(max-min)的倒数
matrix_r0c0	int32	CSC矩阵元素
matrix_r0c1	int32	CSC矩阵元素
matrix_r0c2	int32	CSC矩阵元素
matrix_r1c0	int32	CSC矩阵元素
matrix_r1c1	int32	CSC矩阵元素
matrix_r1c2	int32	CSC矩阵元素
matrix_r2c0	int32	CSC矩阵元素

成员	类型	说明
matrix_r2c1	int32	CSC矩阵元素
matrix_r2c2	int32	CSC矩阵元素
output_bias_0	int32	RGB转YUV时的输出偏移
output_bias_1	int32	RGB转YUV时的输出偏移
output_bias_2	int32	RGB转YUV时的输出偏移
input_bias_0	int32	YUV转RGB时的输入偏移
input_bias_1	int32	YUV转RGB时的输入偏移
input_bias_2	int32	YUV转RGB时的输入偏移

