syntax = "proto3";

package tensorflow;

import "tensorflow/core/framework/attr\_value.proto";

import "tensorflow/core/framework/full\_type.proto";

import "tensorflow/core/framework/resource\_handle.proto";

import "tensorflow/core/framework/types.proto";

option cc\_enable\_arenas = true;

option java\_outer\_classname = "OpDefProtos";

option java\_multiple\_files = true;

option java\_package = "org.tensorflow.framework";

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/framework/op\_def\_go\_proto";

// Defines an operation. A NodeDef in a GraphDef specifies an Op by

// using the "op" field which should match the name of a OpDef.

// LINT.IfChange

message OpDef {

// Op names starting with an underscore are reserved for internal use.

// Names should be CamelCase and match the regexp "[A-Z][a-zA-Z0-9>\_]\*".

string name = 1;

// For describing inputs and outputs.

message ArgDef {

// Name for the input/output. Should match the regexp "[a-z][a-z0-9\_]\*".

string name = 1;

// Human readable description.

string description = 2;

// Describes the type of one or more tensors that are accepted/produced

// by this input/output arg. The only legal combinations are:

// \* For a single tensor: either the "type" field is set or the

// "type\_attr" field is set to the name of an attr with type "type".

// \* For a sequence of tensors with the same type: the "number\_attr"

// field will be set to the name of an attr with type "int", and

// either the "type" or "type\_attr" field will be set as for

// single tensors.

// \* For a sequence of tensors, the "type\_list\_attr" field will be set

// to the name of an attr with type "list(type)".

DataType type = 3;

string type\_attr = 4; // if specified, attr must have type "type"

string number\_attr = 5; // if specified, attr must have type "int"

// If specified, attr must have type "list(type)", and none of

// type, type\_attr, and number\_attr may be specified.

string type\_list\_attr = 6;

// The handle data for resource inputs.

repeated ResourceHandleProto.DtypeAndShape handle\_data = 7;

// For inputs: if true, the inputs are required to be refs.

// By default, inputs can be either refs or non-refs.

// For outputs: if true, outputs are refs, otherwise they are not.

bool is\_ref = 16;

// Experimental. Full type declaration for this argument.

// The full type specification combines type, type\_attr, type\_list\_attr,

// etc. into a unified representation.

// This declaration may contain non-concrete types (for example,

// Tensor<TypeVar<'T'>> is a valid type declaration.

//

// Note: this is a transient field. The long-term aim is to represent the

// entire OpDef as a single type: a callable. In that context, this field is

// just the type of a single argument.

FullTypeDef experimental\_full\_type = 17;

}

// Description of the input(s).

repeated ArgDef input\_arg = 2;

// Description of the output(s).

repeated ArgDef output\_arg = 3;

// Named control outputs for this operation. Useful only for composite

// operations (i.e. functions) which want to name different control outputs.

repeated string control\_output = 20;

// Description of the graph-construction-time configuration of this

// Op. That is to say, this describes the attr fields that will

// be specified in the NodeDef.

message AttrDef {

// A descriptive name for the argument. May be used, e.g. by the

// Python client, as a keyword argument name, and so should match

// the regexp "[a-z][a-z0-9\_]+".

string name = 1;

// One of the type names from attr\_value.proto ("string", "list(string)",

// "int", etc.).

string type = 2;

// A reasonable default for this attribute if the user does not supply

// a value. If not specified, the user must supply a value.

AttrValue default\_value = 3;

// Human-readable description.

string description = 4;

// TODO(josh11b): bool is\_optional?

// --- Constraints ---

// These constraints are only in effect if specified. Default is no

// constraints.

// For type == "int", this is a minimum value. For "list(\_\_\_)"

// types, this is the minimum length.

bool has\_minimum = 5;

int64 minimum = 6;

// The set of allowed values. Has type that is the "list" version

// of the "type" field above (uses the "list" field of AttrValue).

// If type == "type" or "list(type)" above, then the "type" field

// of "allowed\_values.list" has the set of allowed DataTypes.

// If type == "string" or "list(string)", then the "s" field of

// "allowed\_values.list" has the set of allowed strings.

AttrValue allowed\_values = 7;

}

repeated AttrDef attr = 4;

// Optional deprecation based on GraphDef versions.

OpDeprecation deprecation = 8;

// One-line human-readable description of what the Op does.

string summary = 5;

// Additional, longer human-readable description of what the Op does.

string description = 6;

// -------------------------------------------------------------------------

// Which optimizations this operation can participate in.

// True if the operation is commutative ("op(a,b) == op(b,a)" for all inputs)

bool is\_commutative = 18;

// If is\_aggregate is true, then this operation accepts N >= 2

// inputs and produces 1 output all of the same type. Should be

// associative and commutative, and produce output with the same

// shape as the input. The optimizer may replace an aggregate op

// taking input from multiple devices with a tree of aggregate ops

// that aggregate locally within each device (and possibly within

// groups of nearby devices) before communicating.

// TODO(josh11b): Implement that optimization.

bool is\_aggregate = 16; // for things like add

// Other optimizations go here, like

// can\_alias\_input, rewrite\_when\_output\_unused, partitioning\_strategy, etc.

// -------------------------------------------------------------------------

// Optimization constraints.

// Ops are marked as stateful if their behavior depends on some state beyond

// their input tensors (e.g. variable reading op) or if they have

// a side-effect (e.g. printing or asserting ops). Equivalently, stateless ops

// must always produce the same output for the same input and have

// no side-effects.

//

// By default Ops may be moved between devices. Stateful ops should

// either not be moved, or should only be moved if that state can also

// be moved (e.g. via some sort of save / restore).

// Stateful ops are guaranteed to never be optimized away by Common

// Subexpression Elimination (CSE).

bool is\_stateful = 17; // for things like variables, queue

// -------------------------------------------------------------------------

// Non-standard options.

// By default, all inputs to an Op must be initialized Tensors. Ops

// that may initialize tensors for the first time should set this

// field to true, to allow the Op to take an uninitialized Tensor as

// input.

bool allows\_uninitialized\_input = 19; // for Assign, etc.

// Indicates whether the op implementation uses distributed communication.

// If True, the op is allowed to return errors for network disconnection and

// trigger TF network failure handling logics.

bool is\_distributed\_communication = 21;

}

// LINT.ThenChange(

// https://www.tensorflow.org/code/tensorflow/core/framework/op\_def\_util.cc)

// Information about version-dependent deprecation of an op

message OpDeprecation {

// First GraphDef version at which the op is disallowed.

int32 version = 1;

// Explanation of why it was deprecated and what to use instead.

string explanation = 2;

}

// A collection of OpDefs

message OpList {

repeated OpDef op = 1;

}