syntax = "proto3";

package tensorflow;

import "google/protobuf/any.proto";

import "tensorflow/core/framework/tensor\_shape.proto";

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

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

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

import "tensorflow/core/protobuf/struct.proto";

import "tensorflow/core/protobuf/trackable\_object\_graph.proto";

option cc\_enable\_arenas = true;

option go\_package = "github.com/tensorflow/tensorflow/tensorflow/go/core/protobuf/for\_core\_protos\_go\_proto";

// A SavedObjectGraph is part of object-based SavedModels in TF 2.0. It

// describes the directed graph of Python objects (or equivalent in other

// languages) that make up a model, with nodes[0] at the root.

// SavedObjectGraph shares some structure with TrackableObjectGraph, but

// SavedObjectGraph belongs to the MetaGraph and contains pointers to functions

// and type information, while TrackableObjectGraph lives in the checkpoint

// and contains pointers only to variable values.

message SavedObjectGraph {

// Flattened list of objects in the object graph.

//

// The position of the object in this list indicates its id.

// Nodes[0] is considered the root node.

repeated SavedObject nodes = 1;

// Information about captures and output structures in concrete functions.

// Referenced from SavedBareConcreteFunction and SavedFunction.

map<string, SavedConcreteFunction> concrete\_functions = 2;

}

message SavedObject {

// Objects which this object depends on: named edges in the dependency

// graph.

//

// Note: All kinds of SavedObject may have children, except

// "constant" and "captured\_tensor".

repeated TrackableObjectGraph.TrackableObject.ObjectReference children = 1;

// Ordered list of dependencies that must be loaded before this object.

// SavedModel loads with the bottom-up approach, by first creating all objects

// (in the order defined by the dependencies), then connecting the edges.

repeated TrackableObjectGraph.TrackableObject.ObjectReference dependencies =

15;

// Removed when forking SavedObject from TrackableObjectGraph.

reserved "attributes";

reserved 2;

// Slot variables owned by this object. This describes the three-way

// (optimizer, variable, slot variable) relationship; none of the three

// depend on the others directly.

//

// Note: currently only valid if kind == "user\_object".

repeated TrackableObjectGraph.TrackableObject.SlotVariableReference

slot\_variables = 3;

oneof kind {

SavedUserObject user\_object = 4;

SavedAsset asset = 5;

SavedFunction function = 6;

SavedVariable variable = 7;

SavedBareConcreteFunction bare\_concrete\_function = 8;

SavedConstant constant = 9;

SavedResource resource = 10;

CapturedTensor captured\_tensor = 12;

}

// Stores the functions used to save and restore this object. At most one of

// `saveable\_objects` or `registered\_saver` is defined for each SavedObject.

// See the comment below for the difference between SaveableObject and

// registered savers.

map<string, SaveableObject> saveable\_objects = 11;

// The fields below are filled when the user serializes a registered Trackable

// class or an object with a registered saver function.

//

// Registered classes may save additional metadata and supersede the

// default loading process where nodes are recreated from the proto.

// If the registered class cannot be found, then the object will load as one

// one of the default trackable objects: Autotrackable (a class similar to

// tf.Module), tf.function, or tf.Variable.

//

// Unlike SaveableObjects, which store the functions for saving and restoring

// from tensors, registered savers allow Trackables to write checkpoint shards

// directly (e.g. for performance or coordination reasons).

// \*All registered savers must be available when loading the SavedModel.\*

// The name of the registered class of the form "{package}.{class\_name}".

// This field is used to search for the registered class at loading time.

string registered\_name = 13;

// The user-generated proto storing metadata for this object, to be passed to

// the registered classes's \_deserialize\_from\_proto method when this object is

// loaded from the SavedModel.

google.protobuf.Any serialized\_user\_proto = 14;

// String name of the registered saver. At most one of `saveable\_objects` or

// `registered\_saver` is defined for each SavedObject.

string registered\_saver = 16;

}

// A SavedUserObject is an object (in the object-oriented language of the

// TensorFlow program) of some user- or framework-defined class other than

// those handled specifically by the other kinds of SavedObjects.

//

// This object cannot be evaluated as a tensor, and therefore cannot be bound

// to an input of a function.

message SavedUserObject {

// Corresponds to a registration of the type to use in the loading program.

string identifier = 1;

// Version information from the producer of this SavedUserObject.

VersionDef version = 2;

// Metadata for deserializing this object.

//

// Deprecated! At the time of deprecation, Keras was the only user of this

// field, and its saving and loading code will be updated shortly.

// Please save your application-specific metadata to a separate file.

string metadata = 3 [deprecated = true];

}

// A SavedAsset points to an asset in the MetaGraph.

//

// When bound to a function this object evaluates to a tensor with the absolute

// filename. Users should not depend on a particular part of the filename to

// remain stable (e.g. basename could be changed).

message SavedAsset {

// Index into `MetaGraphDef.asset\_file\_def[]` that describes the Asset.

//

// Only the field `AssetFileDef.filename` is used. Other fields, such as

// `AssetFileDef.tensor\_info`, MUST be ignored.

int32 asset\_file\_def\_index = 1;

}

// A function with multiple signatures, possibly with non-Tensor arguments.

message SavedFunction {

repeated string concrete\_functions = 1;

FunctionSpec function\_spec = 2;

}

message CapturedTensor {

// Name of captured tensor

string name = 1;

// Name of concrete function which contains the computed graph tensor.

string concrete\_function = 2;

}

// Stores low-level information about a concrete function. Referenced in either

// a SavedFunction or a SavedBareConcreteFunction.

message SavedConcreteFunction {

repeated int32 bound\_inputs = 2;

// Input in canonicalized form that was received to create this concrete

// function.

StructuredValue canonicalized\_input\_signature = 3;

// Output that was the return value of this function after replacing all

// Tensors with TensorSpecs. This can be an arbitrary nested function and will

// be used to reconstruct the full structure from pure tensors.

StructuredValue output\_signature = 4;

}

message SavedBareConcreteFunction {

// Identifies a SavedConcreteFunction.

string concrete\_function\_name = 1;

// A sequence of unique strings, one per Tensor argument.

repeated string argument\_keywords = 2;

// The prefix of `argument\_keywords` which may be identified by position.

int64 allowed\_positional\_arguments = 3;

// The spec of the function that this ConcreteFunction is traced from. This

// allows the ConcreteFunction to be called with nest structure inputs. This

// field may not be populated. If this field is absent, the concrete function

// can only be called with flat inputs.

// TODO(b/169361281): support calling saved ConcreteFunction with structured

// inputs in C++ SavedModel API.

FunctionSpec function\_spec = 4;

}

message SavedConstant {

// An Operation name for a ConstantOp in this SavedObjectGraph's MetaGraph.

string operation = 1;

}

// Represents a Variable that is initialized by loading the contents from the

// checkpoint.

message SavedVariable {

DataType dtype = 1;

TensorShapeProto shape = 2;

bool trainable = 3;

VariableSynchronization synchronization = 4;

VariableAggregation aggregation = 5;

string name = 6;

string device = 7;

// List of component variables for a distributed variable.

//

// When this field is non-empty, the SavedVariable will be assumed

// to be a distributed variable defined by the components listed here.

//

// This is only supported by experimental loaders at the moment.

repeated SavedVariable experimental\_distributed\_variable\_components = 8;

}

// Represents `FunctionSpec` used in `Function`. This represents a

// function that has been wrapped as a TensorFlow `Function`.

message FunctionSpec {

// Full arg spec from inspect.getfullargspec().

StructuredValue fullargspec = 1;

// Whether this represents a class method.

bool is\_method = 2;

// The input signature, if specified.

StructuredValue input\_signature = 5;

// Whether the function should be compiled by XLA.

//

// The public interface to `tf.function` uses an optional boolean to

// represent three distinct states for this field. Unfortunately, proto3

// removes the ability to explicitly check for the presence or absence of a

// field, so we instead map to an enum.

//

// See `tf.function` for details.

enum JitCompile {

DEFAULT = 0;

ON = 1;

OFF = 2;

}

JitCompile jit\_compile = 6;

reserved 3, 4;

}

// A SavedResource represents a TF object that holds state during its lifetime.

// An object of this type can have a reference to a:

// create\_resource() and an initialize() function.

message SavedResource {

// A device specification indicating a required placement for the resource

// creation function, e.g. "CPU". An empty string allows the user to select a

// device.

string device = 1;

}

message SaveableObject {

// Node ids of concrete functions for saving and loading from a checkpoint.

// These functions save and restore directly from tensors.

int32 save\_function = 2;

int32 restore\_function = 3;

}