

## Control Dependencies

In some cases, it may be wise to postpone the evaluation of an operation even though all the operations it depends on have been executed. For example, if it uses a lot of memory but its value is needed only much further in the graph, it would be best to evaluate it at the last moment to avoid needlessly occupying RAM that other operations may need. Another example is a set of operations that depend on data located outside of the device. If they all run at the same time, they may saturate the device's communication bandwidth, and they will end up all waiting on I/O. Other operations that need to communicate data will also be blocked. It would be preferable to execute these communication-heavy operations sequentially, allowing the device to perform other operations in parallel.

To postpone evaluation of some nodes, a simple solution is to add *control dependencies*. For example, the following code tells TensorFlow to evaluate `x` and `y` only after `a` and `b` have been evaluated:

```
a = tf.constant(1.0)
b = a + 2.0

with tf.control_dependencies([a, b]):
    x = tf.constant(3.0)
    y = tf.constant(4.0)

z = x + y
```

Obviously, since `z` depends on `x` and `y`, evaluating `z` also implies waiting for `a` and `b` to be evaluated, even though it is not explicitly in the `control_dependencies()` block. Also, since `b` depends on `a`, we could simplify the preceding code by just creating a control dependency on `[b]` instead of `[a, b]`, but in some cases “explicit is better than implicit.”

Great! Now you know:

- How to place operations on multiple devices in any way you please
- How these operations get executed in parallel
- How to create control dependencies to optimize parallel execution

It's time to distribute computations across multiple servers!

## Multiple Devices Across Multiple Servers

To run a graph across multiple servers, you first need to define a *cluster*. A cluster is composed of one or more TensorFlow servers, called *tasks*, typically spread across several machines (see [Figure 12-6](#)). Each task belongs to a *job*. A job is just a named group of tasks that typically have a common role, such as keeping track of the model