Ibis Users Guide

The Ibis Group

November 7, 2007

1 Introduction

This manual describes the steps required to run an application that uses the Ibis communication library. How to create such an application is described in the IPL Programmers manual.

A central concept in Ibis is the *Pool*. A Pool consists of one or more Ibis instances, usually running on different machines. Each pool is generally made up of Ibisses running a single distributed applications. Ibisses in a pool can communicate with each other, and, using the registry mechanism present in Ibis, can search for other Ibisses in the same pool, get notified of Ibisses joining the pool, etc. To coordinate Ibis pools a so-called *Ibis Server* is used.

2 The Ibis Server

The Ibis Server is the Swiss-army-knife server of the Ibis project. Services can be dynamically added to the server. By default, the Ibis communication library comes with a registry service. This registry service keeps track of pools, and can track multiple pools at the same time. The server also allows Ibisses to route traffic over the server if no direct connection is possible between two instances due to firewalls or NAT boxes.

The Ibis server is started with the `ibis-server` script which is located in the Ibis `bin` directory. Before starting an Ibis application, an Ibis server needs to be running on a machine that is accessible from all nodes participating in the Ibis run. The server listens to a TCP port. The port number can be specified using the `--port` command line option to the `ibis-server` script. For a complete list of all options, use the `--help` option of the script. One usefull options is the `--event` option, which makes the registry print out events (such as ibisses joining a pool).

3 Running an Ibis Application

When the Ibis Server is running, the Ibis application can now be started. There are a number of requirements if Ibis is to function correctly. We will discuss these in detail below.

3.1 Add ipl.jar to the classpath

An application interfaces to Ibis using the Ibis Portability Layer, or *IPL*. The code for this package is provided in a single jar file: ipl.jar (appended with the version of ibis, for instance `ipl-2.0.jar`. This jar file needs to be added to the classpath of the application.

3.2 Provide the Ibis implementations

The IPL loads the Ibis implementation used to actually communicate dynamically. These implementations (and their dependancies) can be provided in two alternative ways:

1. add the jar files of the implementations and their dependancies to the classpath
2. set the `ibis.implementation.path` system property to the location of the Ibis implementations and dependancies.

System properties can be set in Java using the `-D` option of the `java` command. The ibis.implementation.path property is a list of directories, seperated by the default path seperator of your operating system. In Unix, this is the *;* character, in Windows it is a *:*.

3.3 Configure Log4j

Ibis uses the Log4J library of the Apache project to print debugging information, warnings, and error messages. This library must be initialized. A configuration file can be specified using the `log4j.configuration` system property. For example, to use a file named `log4j.properties` in the current directory, set `log4j.configuration` to `file:log4j.properties`

3.4 Set the location of the Server

To communicate with the registry service, the address of the Ibis server is needed. It must be specified using the `ibis.server.address` system property. The full address needed is printed on startup of the Ibis server.

For convenience, it is also possible to only provide a adress, port number pair, e.g. `machine.domain.com:5435` or even simply a host, e.g. `localhost`. In this case, the port number is implied (the default is 8888). The port number provided must match the one given to ibis-server on startup using the `--port` option.

The Ibis Server is a single point which needs to be reachable from every Ibis instance. Since sometimes this is not possible due to firewalls, additional *hubs* can be started, creating a routing infrastructure for the Ibis instances. These hubs can be started by using ibis-server script with the `--hub-only` option. See the `--help` option of the ibis-server script for more information.

When additional hubs are started, these must also be given to the ibis instances. This can be done using the using the `ibis.hub.addresses` property. Ibis expects a comma seperated list of addresses of hubs.

3.5 Set the properties of the pool

Each pool in Ibis has a unique name. A server can service multiple concurrent pools, and this name is used to determine which ibis instances belong to which pool at the server. The pool name can be set using the `ibis.pool.name` system property.

Sometimes, pools have a fixed size. In these so-called *closed world* pools, the number of ibises in the pool is also needed for ibis to function correctly. The size must be set using the `ibis.pool.size` property. This property is normally not needed, and Ibis will print an error when it requires this property.

4 The ibis-run script

To make it slightly easier to run a Ibis application, a `ibis-run` script is provided with the ibis distribution. The `ibis-run` script can be used as follows

`ibis-run` *java-flags class params*

This script performs the first three steps needed to run an application using Ibis. It adds the ipl.jar and all Ibis implementations to the classpath, and configures log4j. It then runs `java` with any command line options given to it. So, any additional options for Java, the main class and any application parameters can be provided as if `java` was called directly.

The `ibis-run` script needs the location of the Ibis distrobution. This must be provided using the IBIS_HOME environment variable.

5 Example

To illustrate running an Ibis application we will use a simple Hello world application. This application is started twice on a single machine. One instance will send a small message to the other, which will print it.

5.1 Compiling the example

Some example applications for the Ibis communication library are provided with the Ibis distribution, in the `examples` directory. They can be compiled using `ant`, the make-like system for Java. Running `ant` in the examples directory should compile these applications.

Alternatively, they can be compiled using only javac. The sources are located in the `src` directory of the examples. Be sure to add `ipl.jar` from the `lib` directory of the distribution to the classpath.

5.2 Running the example

We will now run the example. All code below assumes the IBIS_HOME environment variable is set to the location of the Ibis distribution.

First, we need a ibis-server. Start a shell and run the `ibis-server` script:

```
$ $IBIS_HOME/bin/ibis-server --events
```

As you can see, we provided the `--events` option to get some extra information on when ibises join and leave the pool.

Next, we will start the application twice. One instance will act as the "server", and one the "client". The application will determine who is the server and who is the client automatically. Therefore we can simply start the application using the same command line. In two different shells type:

```
$ CLASSPATH=$IBIS_HOME/examples/lib/ipl-examples-2.0.jar $IBIS_HOME/bin/ibis-run \
    -Dibis.server.address=localhost -Dibis.pool.name=test \
    ibis.ipl.examples.Hello
```

This sets the CLASSPATH environment variable to the jar file of the application, and called ibis-run. Now, you should have the two running instances of your application. One of them should print:

```
 Server received: Hi there
```

As said, the ibis-run script is mearly provided for convenience. To run the application without ibis-run, the following command can be used:

```
$ java \
    -cp
    $IBIS_HOME/lib/ipl-2.0.jar:$IBIS_HOME/examples/lib/ipl-examples-2.0.jar \
    -Dibis.impl.path=$IBIS_HOME/lib \
    -Dibis.server.address=localhost \
    -Dibis.pool.name=text \
    -Dlog4j.configuration=file:$IBIS_HOME/log4j.properties \
    ibis.ipl.examples.hello.Hello
```

In this case, we use the `ibis.impl.path` property to supply Ibis with the jar files of the Ibis implementations. Alternatively, they could also all be added to the classpath.

6 Further Reading

The Ibis web page `http://www.cs.vu.nl/ibis` lists all the documentation and software available for Ibis, including papers, and slides of presentations.

For detailed information on developing an Ibis application see the Programmers Manual, available in the docs directory of the Ibis distribution