Connecting to an RBNB DataTurbine using Python

1. Overview

The RBNB DataTurbine is written in Java and as such provides a Java API for interacting with the server. The Python language is interpreted to byte code prior to execution, and the Python Project supports three different interpreters:

- CPython - the standard C-based Python interpreter
- Jython - the Java-based Python interpreter
- Python.NET - the .NET-based Python interpreter

In order to make calls to the DataTurbine Java API from Python code, the code must be run using the Jython interpreter. This allows standard java libraries to be imported into the python module and used directly in the Python code.

This document covers the software dependencies needed to interact with the DataTurbine and gives an example of a TChainSource class that loads tchain data into the DataTurbine.

2. Dependencies

2.1 Download and install Java

If you don't already have Java on your system, download it from http://java.sun.com. The latest release of Java SE 1.6.21 should work fine. Double click on the downloaded .exe file to install it.

2.2 Download and install Jython

The Jython interpreter runs on Mac OS X, Windows, and Linux, and can be downloaded from http://jython.org/. The current stable version is 2.5.1, and so your python code should be written with this in mind. Double click on the downloaded .jar file to install it.

2.3 Download and install DataTurbine

The RBNB DataTurbine software runs on Mac OS X, Windows, and Linux, and can be downloaded from http://code.google.com/p/dataturbine. The current version is 3.2b4. Double click on the downloaded .jar file to install it.

2.4 Download the BBL software repository

The python tchain example driver is maintained in the BBL software repository located at https://bbl.ancl.hawaii.edu/projects/bbl. You can use a Subversion client to checkout a working copy of the source code, or you can download a snapshot of the code as a zip file from https://bbl.ancl.hawaii.edu/projects /bbl/trunk/kilonalu-0.5.0.zip. Unzip the .zip file in the location of your choice.

3. TChainSource Example

3.1 Overview

The TChainSource driver is a bare-bones example of an instrument driver written in Python with select RBNB DataTurbine libraries imported into it to support calls to the DataTurbine. There is no exception handling, external configuration, or logging in order to keep the code brief. The tchain python module is within the 'instruments' package and the 'tchain' subpackage, but can be called standalone as well.

The tchain module can be run from the command line using the Jython interpreter. The general flow of execution is:

- tchain.py is run via Jython
- tchain.main() is called, which sets up initial variables
- Using the set variables, tchain.main() creates an instance of TChainSource
- TChainSource.__init__() sets the class attributes, creates a connection to the DataTurbine, and registers the data source and its channels
- execution is returned to tchain.main(), and it calls TChainSource.start() to start the data streaming.
- TChainSource.start() calls TChainsource.connect() to get a connection to the instrument. In the example, it opens a data file rather than a TCP socket
- TChainSource.start() iterates through the lines of the file and calls TChainSource.insertData(), sending it a sample to insert
- TChainSource.insertData() creates a timestamp, appends it to the sample string, and puts both the timestamp and the sample into a ChannelMap, which is the data structure used to push data into the DataTurbine as well as pull it out. The ChannelMap is then inserted into the DataTurbine using Source.Flush().
- Once all data lines are inserted, TChainSource.stop() is called, which in turn closes the data file with file.Close() and closes the DataTurbine connection with Source.Detach().
- With the connections closed, execution is returned to the tchain module and sys.exit() is called.

Note that the Java DataTurbine API calls are limited to:

1. TChainSource.__init__() for the connection and registration
2. TChainSource.insertData() to insert the data
3. TChainSource.stop() to close the DT connection

3.2 Start the DataTurbine

Open a terminal (Mac or Linux) or a Command Prompt (Windows) and make a directory to store the ring buffered data, change directories to the RBNB installation program directory, and start the server. For instance:

For Mac:

```
mkdir /tmp/rbnb
cd /Applications/RBNB/V3.2B4/bin
java -jar rbnb.jar -H /tmp/rbnb -a localhost:3333
```

For Windows:

```
mkdir C:\temp\rbnb
cd "C:\Program Files\RBNB\V3.2B4\bin"
```

```
java -jar rbnb.jar -H C:\temp\rbnb -a localhost:3333
```

For Linux:

```
mkdir /tmp/rbnb
cd /usr/local/RBNB/V3.2B4/bin
java -jar rbnb.jar -H /tmp/rbnb -a localhost:3333
```

The server will start up both Apache Tomcat and the DataTurbine, and will wait for connections.

3.4 Run the TChainSource driver

Open another Terminal or Command Prompt and change directories to the tchain source code subpackage folder. Then, execute the tchain.py script using Jython. For instance:

For Mac:

```
cd /Users/cjones/bbl/trunk/src/python/instruments/tchain
/Applications/jython2.5.1/jython tchain.py
```

For Windows:

```
cd C:\Users\cjones\bbl\trunk\src\python\instruments\tchain
C:\Program Files\jython2.5.1\jython tchain.py
```

For Linux:

```
cd /home/cjones/bbl/trunk/src/python/instruments/tchain
/usr/local/jython2.5.1/jython tchain.py
```

The Python code should run, connect to the DataTurbine, insert each sample line, and then exit cleanly. It is configured by default to load data from a data file with only ten lines. If you have any trouble, send me a note at csjones@hawaii.edu.