

Wikiprint Book

Title: net-iocache

Subject: Java Path Finder - projects/net-iocache

Version: 17

Date: 03/07/2013 11:02:32 PM

Table of Contents

net-iocache	3
What is net-iocache?	3
How to build.	3
Installation	3
Running net-iocache.	3
Running the examples.	4
Other options/tuning.	4
Jar archives	4

net-iocache

A JPF extension for model checking networked programs, by Watcharin Leungwattanakit and Cyrille Artho.

For any information or to report problems, please contact:

Watcharin Leungwattanakit: watcharin@...

-
- Note: The documentation below refers mostly to the version working for a previous release of JPF, available from sourceforge.net. There now exists a new version, which you can obtain by cloning its repository (also see [install/repositories](#)):

```
hg clone http://babelfish.arc.nasa.gov/hg/jpf/jpf-net-iocache
```

The new version comes with a `jpf.properties` file for configuration with the new version of JPF, and automatically generates a `~/jpf/site.properties` file if necessary. Please refer to the README file for details. The documentation below refers to the configuration using the previous release from JPF:

What is net-iocache?

Net-iocache is a Java PathFinder extension for verifying networked applications. JPF executes one of the processes in an application, whereas the others run on the standard Java virtual machine. Net-iocache captures inputs/outputs of the target program and replays them when the program needs again after backtracking. The algorithm to be run in net-iocache is presented in

- C. Artho, W. Leungwattanakit, M. Hagiya, Y. Tanabe, M. Yamamoto. Cache-Based Model Checking of Networked Applications: From Linear to Branching Time. ASE 2009, November 2009, Auckland, New Zealand.
- C. Artho, W. Leungwattanakit, M. Hagiya, and Y. Tanabe. Efficient model checking of networked applications. In Proc. TOOLS EUROPE 2008, volume 19 of LNBIP, pages 22–40, Zurich, Switzerland, 2008. Springer.

Currently, programs are assumed to be either running as a server, accepting connections from several clients, or as a client, connecting to one server. The algorithm has been tested for depth-first search. Other search types, and other types of protocols (such as peer-to-peer) are not supported yet.

How to build.

You can either build Java PathFinder from the command line with Ant, or from within Eclipse.

To build with Ant, switch to the directory where the net-iocache extension is located (where this file is located), and run

```
ant compile-network
```

which should compile all net-iocache sources including example programs.

Installation

For convenience, a snapshot of the extension (October 1 2009) is attached below. Both .jar files should be copied to `${JPF_HOME}/extensions`.

To generate jar files that can be installed under `${JPF_HOME}/extensions`, run `ant jar-net`. This generates three jar files under `${JPF_HOME}/dist`:

- `net-iocache.jar`: The implementation of the cache; it should be copied to `${JPF_HOME}/extensions`, so it is automatically included in the class path.
- `net-env.jar`: The model classes, which need to be included in the `vm.bootclasspath` of JPF.
- `net-examples.jar`: Examples, which do not have to be installed.

Running net-iocache.

- Note: the current version has been built and tested against JPF v4 (from the sourceforge svn repository). A port to the new version of JPF will simplify the usage of this extension. Until the port is available, please follow the instructions below and use the svn repository or the current stable release from <http://javapathfinder.sourceforge.net/>.

If you want to run your own program, you should set `JPF_HOME` and copy `net-iocache.jar` and `net-env.jar` to `${JPF_HOME}/extensions`. You then need to specify the following options to JPF to use net-iocache:

```
${JPF_HOME}/bin/jpf \
+vm.bootclasspath="${JPF_HOME}/extensions/net-env.jar" \
+jpf.listener=gov.nasa.jpf.network.listener.CacheNotifier \
+boot.peer.command="[for server:command to start client process]" \
Application [application_args]
```

The first argument ensures that the network model classes of the cache are loaded; the second argument (listener) activates the cache. If a client is verified, `boot.peer.command` can be omitted; if a server is verified, the cache needs to know what kind of process to spawn when the server is waiting for a client to connect.

For other (optional) arguments to net-iocache, see below ("tuning").

Running the examples.

You can find the scripts to run the example programs in directory "bin". You may want to make sure that net-iocache is properly built by running

```
./alphabet-client-mc.sh 2 1
```

which verifies the alphabet client with two threads inside JPF and generates the alphabet server process to handle the client threads.

The scripts that end with "-mc" start JPF verifying an example program and the peer process of the program.

You can write your own script by using one of the provided scripts as a template. All "mc" scripts include file "env.sh", which initializes shell variables used in most scripts to default values. The meaning of each variable is described below.

- `JPF_HOME`: JPF base directory
- `SRC_DIR`: net-iocache base directory
- `LIB_DIR`: directory containing necessary jar files
- `BIN_DIR`: directory containing the startup scripts
- `VM_ARG`: This contains arguments for the Java virtual machine.
- `CLASSPATH`: Classpaths that JPF needs including JPF core classes, model classes, native peer classes, and example program classes.
- `JPF_ARG`: Arguments for JPF. net-iocache uses the JPF default property (default.properties) as the base property. You can change values and add extra options by adding "+<option>=<value>" to this variable.

Other options/tuning.

net-iocache introduces three new options: `exception.simulation`, `main.termination`, and `lazy.connect`.

- `exception.simulation`: If this option is set to true, calls to a network operation will result in either success or failure. Both possibilities are simulated by JPF. (Default: false)
- `main.termination`: If it is true, the main thread will run solo until the end before the other threads start running. This option would improve performance of verification when the main thread does nothing but creating worker threads. (Default: true)
- `lazy.connect`: If it is true, net-iocache will not open a physical connection immediately after the program calls method "connect". (Default: true)

Jar archives

The jar files at the bottom of this page contain builds against the old version of JPF (version 4) from sourceforge.net. Barring serious problems, these are going to be the final releases, to be replaced soon with pre-built archives that work with JPF 5, from this page.