PyAALib-JyAALib - http://jyaalib.sourceforge.net
[20080827 – How To: Profiler Tool?]

by Dr. Romeu Andre' PIERITZ

How to: PROFILER TOOL?

NEW: A new GUI to debug and profile massively parallel applications is proposed and implemented by the AALib-PyAALib-JyAALib Framework.

The Profiler GUI is a standard option available in the ALApplication module to plot a "timeline" of the concurrent simultaneous actions during the runtime of the application. The figure 1 shows the main interactive interface of the GUI. It is composed by a set of HTML pages using special Javascript based on the open source "SIMILE" Project at MIT http://simile.mit.edu/timeline.

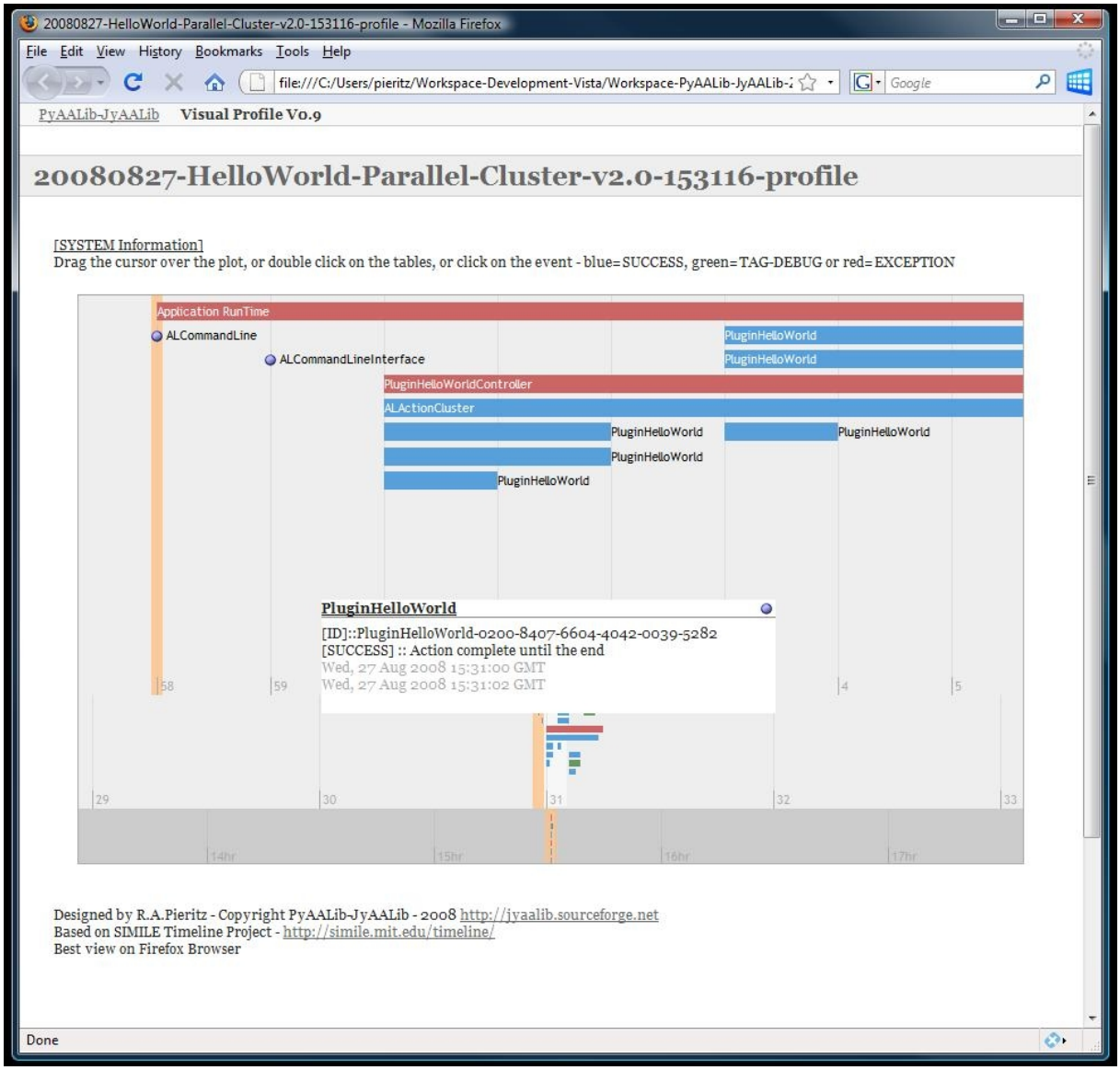

Fig.1: The PyAALib-JyAALib Profiler tool GUI

Why?

It allows an iterative interface to map the asynchronous actions during the runtime of the massively parallel applications. The interface is dynamic, showing Hyperlinks to present the independent output for each dynamic component used during the runtime of the application.

How it works?

The user must add to the command line the option "--profile". It is available for all application based on the AALib ALApplication class. It can be used associated to other commands like DEBUG options (see the documents: Hot to DEBUG? And How to TAG-DEBUG?). The main command line is:

>> myaaplication --profile

or using TAG-DEBUG:
>> myaaplication --TAG-DEBUG myTagPerso --profile

or using the screen output:
>> myaaplication --verbose --profile

Note: "myaaplication" represents a python or Java-Jython script to start the main user code based on PyAALib-JyAALib.

The HTML profile web page is created at the end of the execution of the application. The profiler tool is not consuming resources during the runtime. The profile GUI is composed by a "data base folder", a set of HTML pages using Javascript to plot the timeline. It is compatible with many WEB browsers.

The "Profile HTML File" is identified by the execution time ("xxxxx-HHMMSS-profile.html"), generating one different profile GUI for each execution (different profile file and an independent data base).

Main Features?

The profile tool plots the execution timeline for each independent element based on the main class ALAction (see figure 2).

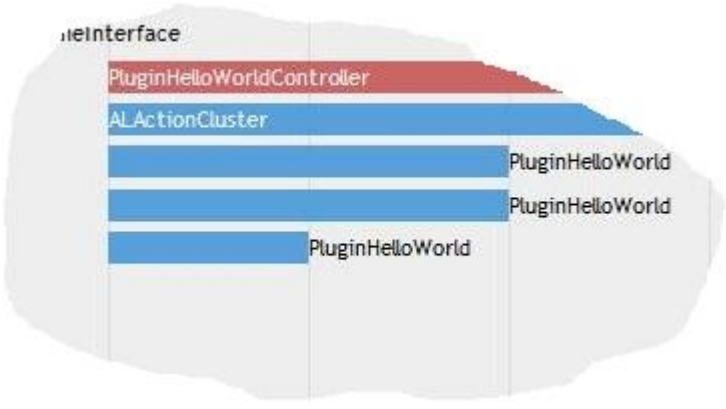

Fig.2: The runtime timeline of dynamic objects and the associated color scheme.

It plots the start time and end time of the dynamic object, showing three main color status:

- BLUE represents a SUCCESS: the object was executed until the end;
- GREEN represents the use of the TAG-DEBUG feature and a SUCCESS;
- RED identify the object when it contains at least one EXCEPTION;

The timeline plot is iterative and dynamic: the mouse drives the timeline in all directions. A single click on the plot move the GUI to the time position.

A single click on each object pop up a dialog box showing the main status and time information - figure 3. The URL link inside the dialog open a log file showing all output from the object code is presented in figure 4.

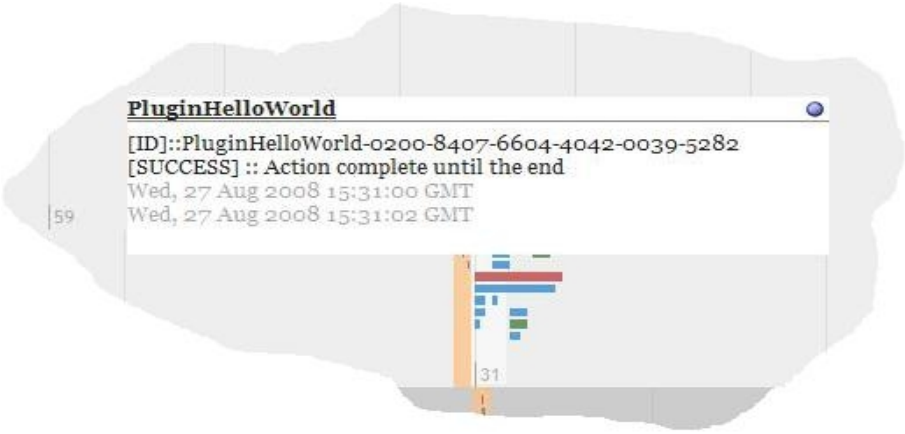

Fig. 3: The pop up dialog box showing the main information of each dynamic object and the Hyperlink to the associated log information.

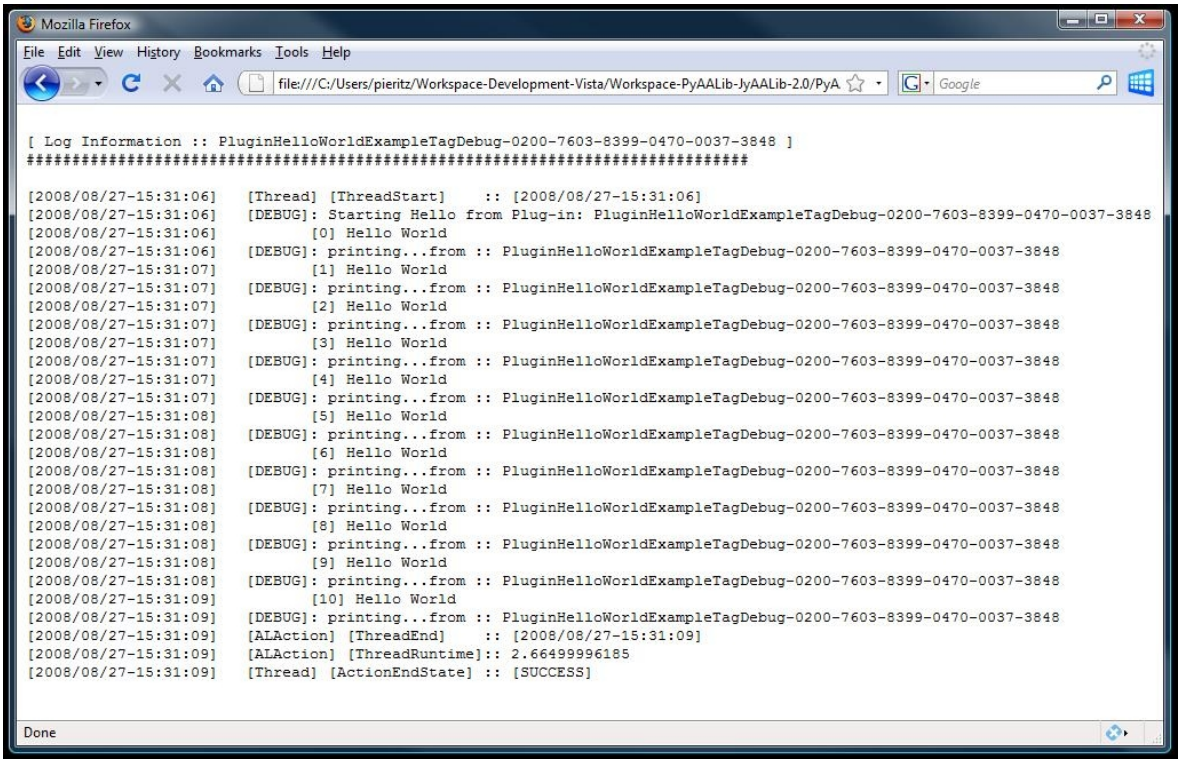

Fig. 4: The log information of a dynamic object using the option TAG-DEBUG.

Different information are available form the GUI, as the runtime options and the log file. In figure 5 is presented the Hyperlink to the main runtime information, presented in the figure 6.

Fig. 5: Special URL for system information is available from the GUI.

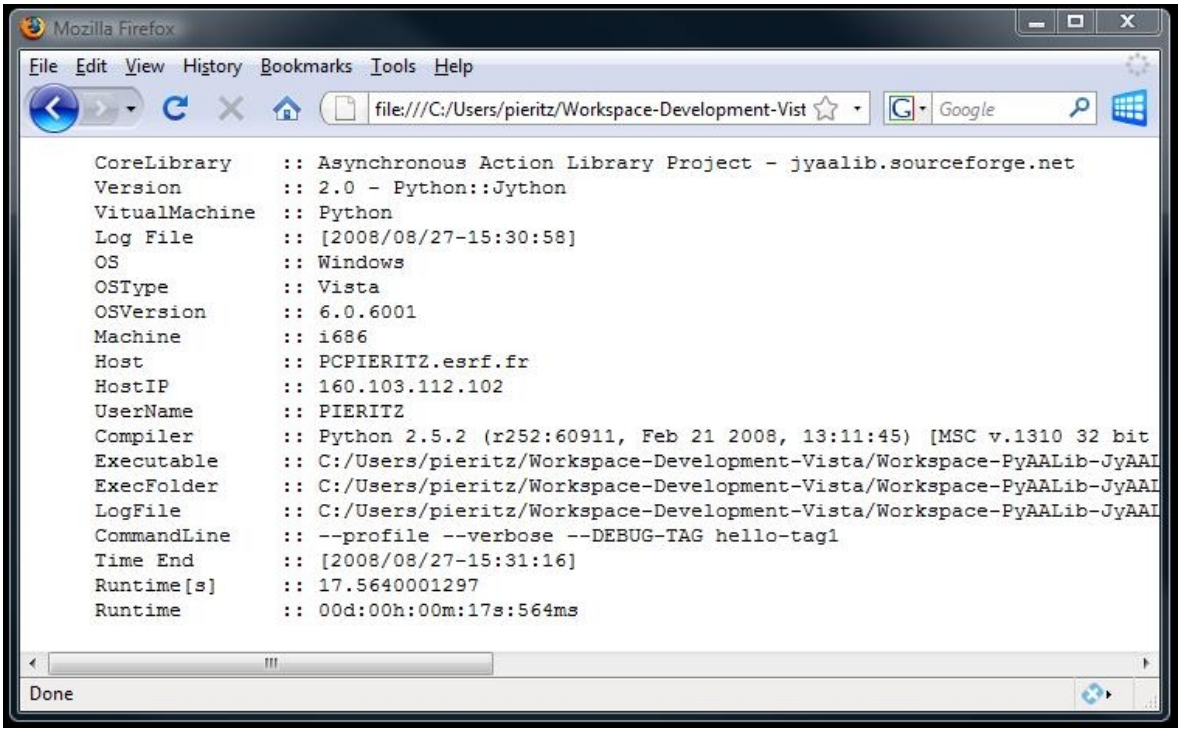

Fig. 6: The runtime system information .

Standard Command Line Options?

The standard built-in options available for all code based on the PyAALib-JyAALib "ALApplication" class are:

--help or -h [option] :	to show help if available
--man or -m [option] :	to show manual if available
--help --all :	it shows the main command list help
--man --all :	it shows the main command list man
--verbose :	to iterative output
--quiet :	NO output (default option)
--no-log :	NO log file output
--profile :	profile output for trace actions
--version or -v :	executable version info
--plugin :	show the plug-in connected to the application
--plugin-group :	show the plug-in groups
--plugin-group-tree :	show the plug-in group tree
--DEBUG :	output debug info with no stops
--DEBUG-TAG [tag] :	output only user Tag debug - all, aalib or others
--DEBUG-FORCE :	output default Python-Java debug info and FORCE the application to stop
--DEBUG-FORCE-PLUGIN :	output default Python-Java debug only from a PLUGIN and FORCE the application to stop
--ASSERT :	output assert info for debug
--import-resource [filename] :	import a file application resource file

Compatible?

It was tested on: Windows Vista32, Ubuntu Linux and ESRF Linux. The MacOSX version is on test.

- On Windows Vista32 and Ubuntu Linux, it was tested using: Python-2.5.2 and Java/Jython-2.2.1.
- On ESRF Linux, the AALib was tested using: Python-2.4.4 and Jython-2.2.

Note: We recommend use the Firefox Web Browser to display the Profile GUI . Some users are experimenting problems with Javascript on Windows Internet Explorer an KDE Web viewer.