Go4: Multitasking Multithreaded Class Library

J. Adamczewski, M. Al-Turany, H. G. Essel, H. Göringer
DVEE, GSI Darmstadt

At the beginning of the GSI Online-Offline-Object-Oriented analysis project GO4, the principle of a multi threaded analysis based on the ROOT framework [1] had already been demonstrated by a first prototype [2]. In the following phase, the software has been redesigned completely using UML tools such as *RationalRose 2000* [3] and, more recently, *Together 4.2* [4], in connection with the *Sniff++* development environment [5].

Go4 Thread Manager

The *Go4ThreadManager* package provides foundation and service classes to launch any number of named threads within a ROOT application. It is based on the ROOT TThread library which had been updated in the course of the first Go4 prototype. The *Go4ThreadManager* implements the concept of runnable classes (like JAVA) which the user may specialize by inheritance for any job without changing the Go4 thread classes themselves.

Go4 Task Handler

The first goal of the Go4 kernel was to control several independent, distributed analysis clients (slaves) from one user interface server task (master). Therefore, the *Go4TaskHandler* package and the related service packages (*Go4Socket*, *Go4Queue*, *Go4CommandsBase*, and *Go4StatusBase*) were designed for such inter task connections.

The client communicates with the server via three sockets (data, command, and status channel). Each of these channels is processed by a dedicated thread and is buffered against the user's application by means of a thread safe template queue. The entire communication setup is encapsulated within the *Go4TaskHandler* class.

Exchange of information between server and clients is done by command objects and by status objects. Here we use a command design pattern [6] with an invoker singleton, and a modified memento pattern, respectively. Commands created by the server may be either sent to one of the remote clients, or added to a local command queue to be executed in a local thread preventing the blocking of the GUI.

Status objects are created by the clients and are sent to the server which may e.g. display the current analysis status. Additionally, any named ROOT object created by an analysis client (e.g. a histogram) can be sent to the GUI via the task handler data channel.

A new client process may either be launched from an existing GUI server process and can be added to the list of clients; or the client may be started independently and may request a new connection to the server process at any time.

Test of Go4 Multitasking

As a first test of the *Go4TaskHandler* package, we built an example client (subclass of TGo4ClientTask), and an example display (subclass of TGo4ServerTask). The client has two additional threads working on an example application (the actual analysis later on) . The client status information is sent to the server regularly by thread one, while thread two executes commands and processes the analysis (here a random histogram fill). The server has a simple GUI control panel, a ROOT canvas and a status window.

Two server threads wait to display any client objects appearing at the status and data queues, respectively. Pressing a GUI button, a histogram is requested from the currently selected client by command, sent to the server and drawn on the canvas. This example was running successfully over >24 hours with 1 server task connected to 7 client tasks on 4 different nodes, which promises a stable operation of the task handler system.

Go4 Viewer

In addition to the design of the Go4 framework, the *Go4Viewer* is being built as a first tool for visualization and interactive manipulation of histograms, ntuples and root TTrees. Based on native ROOT GUI classes it features access to data from different sources: from GSI histogram servers like LeA, GOOSY and MBS, from local root or paw files, and from the GSI mass storage system using the root TRFIO classes. Histograms and ntuples from any data sources are converted on the fly to ROOT file format and may be analyzed later by all means of the ROOT system.

Conclusions and Outlook

The Go4 Task handler package might be a flexible tool for any kind of distributed tasks using the ROOT environment. In contrast to the existing parallel root facility PROOF (which is specialized for parallel processing on the same dataset), the *Go4TaskHandler* is capable of controlling independent clients with threaded applications, required both for non blocking online analysis or slow control jobs.

The next step will be the implementation of the actual analysis framework, containing abstract interfaces for the event related classes, a dynamic list which will keep and process online generated histogram objects, and a system of analysis condition classes. Here we will still benefit from the first Go4 prototype experiences.

The *Go4Viewer* may be a test bed for the future GUI layout, until the first test analysis of the Go4 framework will produce viewable data.

Documentation of Go4 can be found at *http://go4.gsi.de.*

References

[1] http://root.cern.ch

[2] *Status of ROOT Based Analysis System Go4*, J.Adamczewski, H.G.Essel, H.Göringer, M. Hemberger, N. Kurz, M.Richter; GSI scientific report 1999, p.232

[3] http://www.rational-software.de

[4] http://www.togethersoft.com

[5] http://www.windriver.com/products/html/sniff.html

[6] *Design Patterns: Elements of Reusable Object-Oriented Software*, E.Gamma, R.Helm, R.Johnson, J.Vlissides ; Addison-Wesley 1999