GPC – A Job Scheduler (not only) for Radar Image Processing

Hubert Högl
German Aerospace Center / Remote Sensing Data Center (DFD)
Oberpfaffenhofen
D-82234 Weßling
Hubert.Hoegl@dlr.de

Abstract

The Generic Processor Control (GPC) is a portable and lightweight Job Scheduler to coordinate the execution of hundreds of UNIX processes on a single workstation constrained by a set of configurable activation rules. Although originally designed for RADAR image processing in the SRTM project [1][2], the implemented features are fairly generic so that a broad range of applications requiring an extensible Job Scheduler can benefit from GPC. The current prototype is completely implemented using the Python [3] programming language.

Architecture. GPC is a hierarchical and state-based scheduler with a fixed core implementation plus embedded extensions for configuration. The basic units of execution are called *S*cheduled *Objects* (SOB), which are classified into *Blocks* and *Steps*. Both are arranged in a tree datastructure, with the leaves of the tree being exclusively Steps and all other nodes being Blocks.

Steps are equivalent to ordinary UNIX processes while Blocks encapsulate an arbitrary number of Steps and other Blocks. Each Block contains local knowledge how to sequence it's child SOBs. The dynamic behaviour of Steps and Blocks is defined by a state-machine with five states (Init, Run, Sus, Idle, Term) and six states (Init, Run, Wait-for-Stop, Stop, Sus, Term) respectively.

The root Block is called a *Task*. Tasks (including the full SOB tree) are created when a *Processing Request* (PR) enters GPC. After a Task is created it is pushed into a Task Queue where it waits to be automatically or manually selected for execution. After a Task enters the terminated state, it is automatically removed from the Task queue.

For our application a single Task comprises between ten and 800 process executions. These processes operate within a Task working directory tree (workspace). Handling the Task workspace (setup, cleanup, error checks) is also part of GPC's configuration.

PRs contain the parameters for product generation in structured textual form (XML format). GPC extracts a PR's contents, reformats it and places it at predefines places in the Task working directory. Optionally PR information is used to parametrize the structure of the SOB tree.

Configuring GPC means to specify the (a) SOB tree structure, (b) local activation rules of each Block describing how to sequence it's childs, (c) PR readers, (d) Task setup and cleanup procedures, and (e) Task Queue selection policy.

A combination of the following rules can be specified to activate a Block/Step: (a) state of arbitrary number of predecessor SOBs, (b) arrival of arbitrary files, (c) activation time-window and (d) availability of logical resources.

Prototyping with Python. GPC has grown to an 18000 line program within the last eight months (equivalent to four to eight times more code when written in typical system programming languages). Using Python proved to be fast and resulted in code readability and quality impressively higher compared to languages like C or Java. Numerous powerful packages exist (e.g. Tk, XML, CORBA) which boost productivity. Many projects, especially those suffering from permanently tight time pressure and creeping requirement changes, such as our's, can benefit by using Python.

References

[1] *German Remote Sensing Data Center,* `http://www.dfd.dlr.de` *(1998)*

[2] *Shuttle Radar Topography Mission,* `http://www-radar.jpl.nasa.gov/srtm/mission.html` *(1998)*

[3] *Python Home Page,* `http://www.python.org` *(1998)*