Grid Task Execution

Ames Task Execution Center, Moffett Field, California

IPG Execution Service is a framework that reliably executes complex jobs on a computational grid, and is part of the IPG service architecture designed to support location-independent computing. The new grid service enables users to describe the platform on which they need a job to run, which allows the service to locate the desired platform, configure it for the required application, and execute the job. After a job is submitted, users can monitor it through periodic notifications, or through queries.

Each job consists of a set of tasks that performs actions such as executing applications and managing data. Each task is executed based on a starting condition that is an expression of the states of other tasks. This formulation allows tasks to be executed in parallel, and also allows a user to specify tasks to execute when other tasks succeed, fail, or are canceled.

The two core components of the Execution Service are the Task Database, which stores tasks that have been submitted for execution, and the Task Manager, which executes tasks in the proper order, based on the user-specified starting conditions, and avoids overloading local and remote resources while executing tasks.

Parallel-Processing Software for Correlating Stereo Images

NASA’s Jet Propulsion Laboratory, Pasadena, California

A computer program implements parallel-processing algorithms for correlating images of terrain acquired by stereoscopic pairs of digital stereo cameras on an exploratory robotic vehicle (e.g., a Mars rover). Such correlations are used to create three-dimensional computational models of the terrain for navigation. In this program, the scene viewed by the cameras is segmented into subimages. Each subimage is assigned to one of a number of central processing units (CPUs) operating simultaneously. Because each subimage is smaller than a full image, the correlation process takes less time than it would if full images were processed on one CPU. Segmentation and parallelization also make the process more robust in that the smaller subimages present fewer opportunities for a correlation algorithm to “get lost” and thereby fail to converge on a solution. The effectiveness of this program has been demonstrated on several parallel-processing computer systems. Whereas correlation processing of a typical stereoscopic pair of test images on a single CPU was found to take on the order of one hour, parallel processing of the same images on a 16-CPU cluster was found to take about 5 minutes.

Knowledge Base Editor (SharpKBE)

NASA’s Jet Propulsion Laboratory, Pasadena, California

The SharpKBE software provides a graphical user interface environment for domain experts to build and manage knowledge base systems. Knowledge bases can be exported/translated to various target languages automatically, including customizable target languages. The tool enhances current practices by simplifying and parallelizing algorithms for correlating stereo images. In this program, the scene viewed by the cameras is segmented into subimages. Each subimage is assigned to one of a number of central processing units (CPUs) operating simultaneously. Because each subimage is smaller than a full image, the correlation process takes less time than it would if full images were processed on one CPU. Segmentation and parallelization also make the process more robust in that the smaller subimages present fewer opportunities for a correlation algorithm to “get lost” and thereby fail to converge on a solution. The effectiveness of this program has been demonstrated on several parallel-processing computer systems. Whereas correlation processing of a typical stereoscopic pair of test images on a single CPU was found to take on the order of one hour, parallel processing of the same images on a 16-CPU cluster was found to take about 5 minutes.

This program was written by Chaomin Hu of Advanced Management Technology for Ames Research Center. For further information, access http://opensource.arc.nasa.gov/ or contact the Ames Technology Partnerships Division at (650) 604-2954.

ARC-15529-1
C++ targets. The knowledge base, which is created via graphical expression editors, is saved to a standardized XML document structure that is more flexible than previously existing formats, which were written in LISP-style syntax. This provides the user with intuitive mechanisms for viewing and modifying knowledge bases (see example) as well as the ability to produce customized style sheets. The editor can support the auto-coding paradigm for fault detection systems in flight software applications with reduced cost.

This program was written by Raffi Tikidjian, Mark James, and Ryan Mackey of Caltech for NASA’s Jet Propulsion Laboratory. Further information is contained in a TSP (see page 1).

This software is available for commercial licensing. Please contact Karina Edmonds of the California Institute of Technology at (626) 395-2322. Refer to NPO-44476.

**Parallel Computing With Satellite Orbit Analysis Program**

NASA’s Jet Propulsion Laboratory, Pasadena, California

This software innovation speeds up the computation time of the Satellite Orbit Analysis Program (SOAP) tool by parallelizing the code using the message-passing interface (MPI) library. The speed increases almost linearly with the number of processors, allowing the per-study duration of the visualization and analysis of space missions to take place in hours, rather than overnight. The software can conduct a parametric study involving millions of design vectors in a few hours of computational time by distributing the design scenarios among multiple processors. This allows SOAP to run in a parallel mode on JPL’s new, high-performance computer cluster, which has 1024 Intel Xeon processors.

Parametric Study andContours in SOAP were the first targets of parallelization. The two functions compute one or more variables over a given time period and become quickly computationally intensive as report duration, time resolution, and variable complexity increase. The spatial and variable domains are sliced and distributed to each processor proportionally to its processing power. The result of the computation over the slice is collected at the end of the computation, and a single processor handles the file-writing task. Each space and variable domain contribution is completely independent so that message