"Use of Python as a Matlab Replacement for Algorithm Development and Execution in a Multi-Core Environment"

Glen W. Mabey, Southwest Research Institute, San Antonio, TX
Brian Granger, Tech-X Corporation, Boulder, CO

This work represents a case study on the use of the Python programming language and its associated toolkits as a basis for simulation and algorithm development on multicore CPUs and clusters.

Python is a general-purpose, open-source interpreted programming language that supports a variety of programming models and has an extensive set of built in packages. In addition, mature third party packages provide all of the tools needed for serious numerical work, including multidimensional arrays, linear algebra, FFTs and random variates (NumPy) and publication quality 2D plotting (Matplotlib). The broad acceptance of Python in scientific computing is highlighted by the May/June 2007 issue of Computing and Science and Engineering (IEEE) that was entirely devoted to the topic.

Another package available for Python, called IPython, enables parallel and distributed Python applications to be efficiently executed on multicore CPUs and clusters and supercomputers. By supporting many styles of parallelism (task-farming, message passing and other high level approaches), IPython enables many simulations and algorithms to be parallelized quickly and often in a fully interactive manner.

The case study described in this work consists of using Numpy, Matplotlib and IPython to develop a signal processing algorithm and to execute the algorithm in parallel over a large data set. The data consisted of complex samples acquired at a sampling rate of 2.5 MHz. For the duration in use, the acquisition amounted to 1/2 TB of on-disk data. Setting the algorithm itself aside, in the present work we focus on the specific characteristics and advantages of using these tools for this type of work:

- Random access of a data set that is much larger than the available system RAM, without the need to artificially segment the data nor implement direct-disk-access routines.
- A data indexing syntax that is user-friendly and intuitive.
- Availability of common statistical and signal processing operations.
- Access to plotting facilities that are mature enough to allow for rapid visualization of the data and the results, yet at the same time extensible enough to include as part of a full-fledged application, complete with database access routines and custom widgets.
- Rapid development and implementation via a powerful scripting language.
- The option to easily distribute computation of the developed algorithms over a cluster of network-connected nodes so that a large amount of test data may be processed in a reasonable amount of time.

In the execution of the signal processing algorithm mentioned, a cluster of 10 4-core blades was employed, resulting in a 37x speedup.

While there do exist commercial products that provide many of these features, the present approach has the further advantage of being based on freely available open-source software. This allows users to audit, modify, customize and optimize the underlying code and also avoid the expensive costs of licensing the commercial products on multiple CPUs and clusters.