**Tip: Using task parallelism in multicore LabView**

Elizabeth Luley, National Instruments - September 19, 2007

*For more on this topic see Using pipelining in multicore LabView and Using data parallelism in multicore LabView.*

Until recently, advances in computing hardware have provided significant increases in the execution speed of software, with little effort from software developers. Increases in processor speed brought an instant boost in software speed. However, as processor speeds start to peak, and processor manufacturers use new techniques to increase processing power, this is starting to change. The introduction of multicore processors provides a new challenge for software developers, who must now master the programming techniques necessary to capitalize on multicore processing potential. One of these programming techniques is task parallelism.

**Task Parallelism**

Task parallelism is simply the concurrent execution of independent tasks in software. Consider a single-core processor that is running a Web browser and a word-processing program at the same time. Although these applications run on separate threads, they still ultimately share the same processor. Now consider a second scenario in which the same two programs are running on a dual-core processor. On the dual-core machine, these two applications essentially can run independently of one another. Although they may share some resources that prevent them from running completely independently, the dual-core machine can handle the two parallel tasks more efficiently.

The inherent parallelism of dataflow programming makes the National Instruments LabVIEW graphical development environment the ideal programming language for using parallel programming techniques. Traditional text-based languages have a sequential syntax and are therefore difficult to visualize and organize in a parallel form. In contrast, creating a multi-threaded application is intuitive and simple in NI LabVIEW.

**Task Parallelism in LabVIEW**

The LabVIEW graphical programming paradigm makes parallel programming easy, even for novice users. Two separate tasks that are not dependent on one another for data run in parallel without the need for any extra programming.

Figure 1 shows a simple data acquisition routine. The top section of code consists of an analog voltage input task, and the bottom code is a digital output task.
In LabVIEW, two sections of code with no data dependencies will run independently in parallel.

These two independent data acquisition tasks must share the same CPU on a single-core processor. On a multicore processor, each thread can run on its own processor, greatly improving the execution speed.

When using data parallelism in an application, it is important to ensure that the two tasks do not have any shared resources that could create a bottleneck, such as the code in Figure 2.

In this program, the two tasks share data; the digital task is forced to wait until the analog task has already started. Although this code appears to be parallel, it is not truly parallel because the two tasks share data. It is important to be aware of this pitfall when programming applications using task parallelism.

An Application of Task Parallelism
At Eaton Corporation, a team of developers - whose sole mission is to refine the test and measurement systems used in Eaton's truck R&D division - harnessed the performance of multicore processing and the multithreaded architecture of LabVIEW to more than quadruple the number of channels running through Eaton's systems and achieve real-time determinism. This LabVIEW application comprises three asynchronous loops completing the parallel tasks of acquisition, test control, and user interface. Before implementing this solution, the team could only meet its testing requirements by either by dividing the workload sequentially or by using several single-core desktops to run the application. Using standard, off-the-shelf desktop systems, the team was able to minimize power consumption, thermal output, and test time, and thus reduce overall costs.

With task parallelism and other programming techniques, software developers can take full
advantage of the power offered by multicore processing. Developers who are aware of these techniques and understand them are able to prepare applications for the computing trends of the future. The inherent parallel nature of the LabVIEW graphical programming environment gives developers the ability to effectively generate the applications of tomorrow.

For More Information about Multicore Programming, Visit:

- Multicore Fundamentals Series
- Multicore Programming Resources
- Programming Strategies for Multicore Processing: Data Parallelism
- Programming Strategies for Multicore Processing: Pipelining
- Learn More about LabVIEW

About the author

Elizabeth Luley is an Applications Engineer at National Instruments, and co-author of the Multicore Programming Fundamentals technical paper series.