Programming Strategies for Multicore Processing: Data Parallelism

Overview

This document is part of the

Multicore Programming Fundamentals Whitepaper Series

#xD;

Multicore Programming Fundamentals Whitepaper Series

As hardware designers turn toward multicore processors to improve computing power, software programmers must find new programming strategies that harness the power of parallel computing. One technique that effectively takes advantage of multicore processors is data parallelism.

Data Parallelism

Data parallelism is a programming technique for splitting a large data set into smaller chunks that can be operated on in parallel. After the data has been processed, it is combined back into a single data set. With this technique, programmers can modify a process that typically would not be capable of utilizing multicore processing power, so that it can efficiently use all processing power available.

Consider the scenario in Figure 1, which illustrates a large data set being operated on by a single processor. In this scenario, the other three CPU cores available are idle while the first processor solely bears the load of processing the entire data set.

LabVIEW™, National Instruments™, and ni.com™ are trademarks of National Instruments Corporation. Product and company names mentioned herein are trademarks or trade names of their respective companies. For patents covering National Instruments products, refer to the appropriate location: Help»patents in your software, the patents.txt file on your CD, or ni.com/patents.

© Copyright 2006 National Instruments Corporation. All rights reserved.

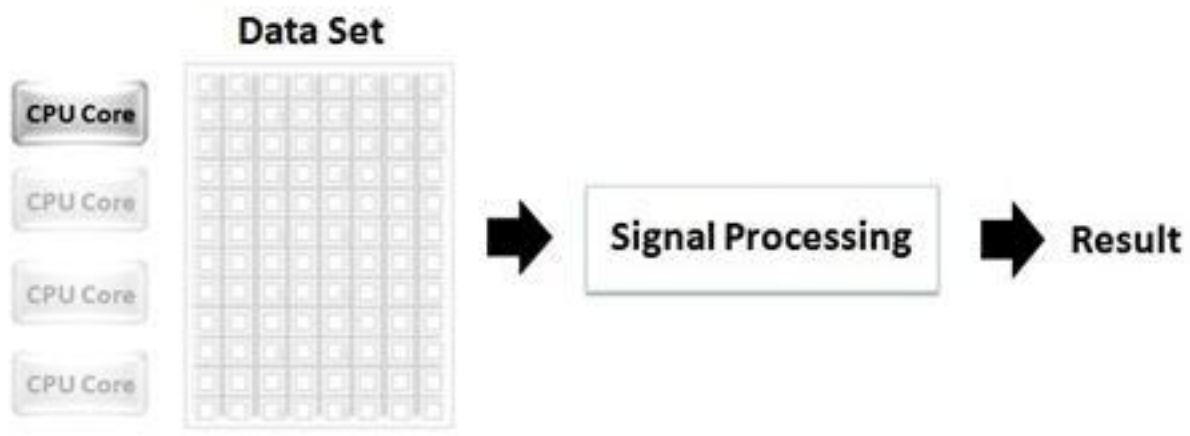

Figure 1. In traditional programming methods, a large data set is processed on a single CPU core, while the other CPU cores remain idle.

Now consider the implementation shown in Figure 2, which uses data parallelism to fully harness the processing power offered by a quad-core processor. In this case, the large data set is broken into four subsets. Each subset is assigned to an individual core for processing. After processing is complete, these subsets are rejoined into a single full data set.

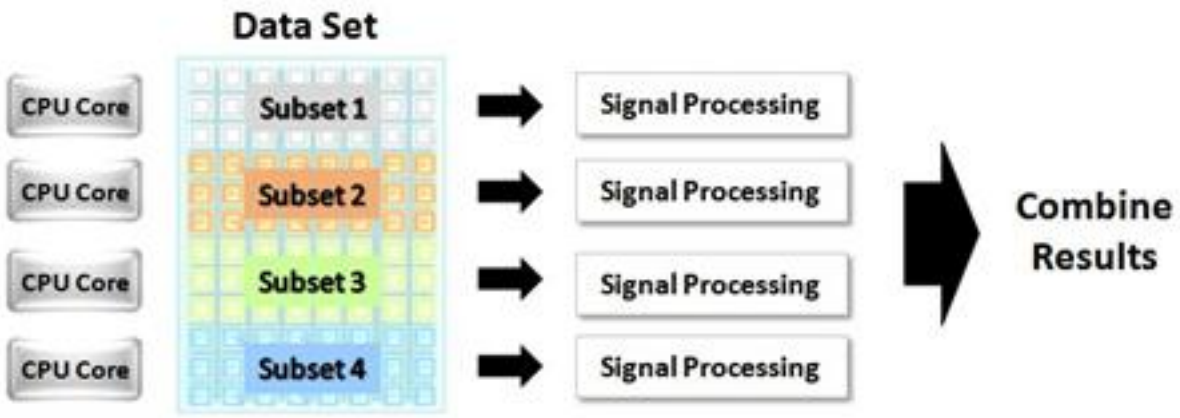

Figure 2. By using the programming technique of data parallelism, a large data set can be processed in parallel on multiple CPU cores.

The graphical programming paradigm of National Instruments LabVIEW is ideal for parallel data architectures. Parallel processing in NI LabVIEW is intuitive and simple, in contrast to traditional text-based languages, which require advanced programming knowledge to create a multithreaded application.

Data Parallelism in LabVIEW

The code in Figure 3 shows a matrix multiply operation performed on two matrices, Matrix 1 and Matrix 2. This is a standard implementation in LabVIEW of multiplication of two large matrices.

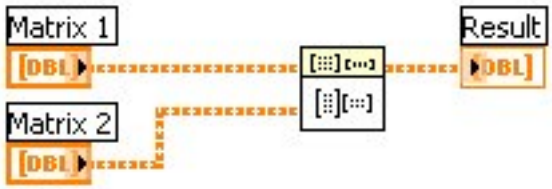

Figure 3. A standard implementation of matrix multiplication in LabVIEW does not use data parallelism.

An operation such as this can take a significant amount of time to complete, especially when large data sets are involved. The code in Figure 3 in no way capitalizes on the extra computer power offered by a multicore processor, unless by chance the Matrix Multiply VI is already multithreaded and fully optimized for a multicore processor. In contrast, the code in Figure 4 makes use of data parallelism and thus can execute significantly faster on a dual-core processor than the code shown in Figure 3.

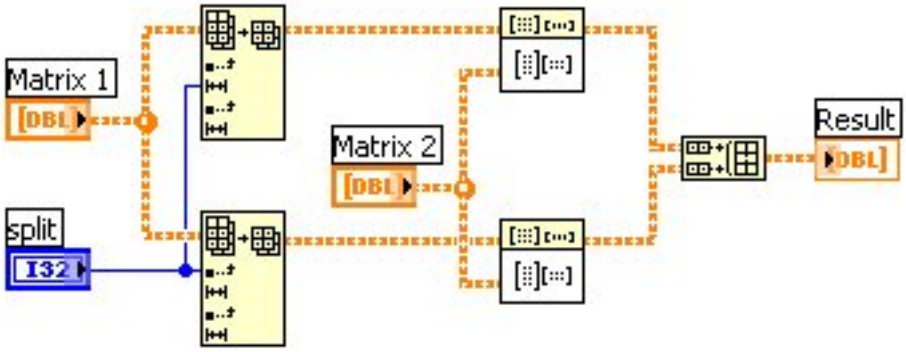

Figure 4. This LabVIEW code demonstrates data parallelism; by dividing the matrix in half the operation can be computed simultaneously on two CPU cores.

When using this technique in LabVIEW to increase performance on a multicore processor, it is important to realize that a performance improvement cannot be achieved unless the Matrix Multiply VI is reentrant. If it is not reentrant, the separate instances of the Matrix Multiply VI cannot operate independently and concurrently.

The table below shows benchmarks for execution time of the LabVIEW code in Figures 3 and 4 with 1000 x 1000 matrices used for the input matrices. A performance improvement is achieved even on the single core

processor when data parallelism is used because the task is more multithreaded. Note that a performance improvement of nearly 2x is seen on the dual core processor when data parallelism is used.

	Execution Time on Single Core Processor	Execution Time on Dual Core Processor
Matrix Multiplication without Data Parallelism	1.195 seconds	1.159 seconds
Matrix Multiplication with Data Parallelism	1.224 seconds	0.629 seconds

An Application of Data Parallelism

At the Max Planck Institute in Munich, Germany, researchers applied data parallelism to a LabVIEW program that performs plasma control of Germany's most advanced nuclear fusion platform, the ASDEX tokamak. The program runs on an octal-core server, performing computationally intensive matrix operations in parallel on the eight CPU cores to maintain a 1 ms control loop. Lead researcher Louis Giannone notes, "In the first design stage of our control application programmed with LabVIEW, we have obtained a 20X processing speed-up on an octal-core processor machine over a single-core processor, while reaching our 1 ms control loop rate requirement."

With data parallelism and similar techniques, programmers can fully exploit multicore processing power. Restructuring processes that are not inherently parallel to a parallel form helps software architects realize the full potential of multicore processors, while LabVIEW easily represents the parallelism in the code.

More Resources on Multicore Programming

Multicore Programming Fundamentals Whitepaper Series

- www.ni.com/multicore
- Programming Strategies for Multicore Processing: Task Parallelism
- Programming Strategies for Multicore Processing: Pipelining
- Learn more about NI LabVIEW