Parallel Functionality and Automatic Thread Control in PV-WAVE® 10.0

PV-WAVE Performance Improvement Brochure

Rogue Wave Software

April 2012

Rogue Wave Software
5500 Flatiron Parkway,
Suite 200
Boulder, CO 80301, USA
www.roguewave.com

Parallel Functionality and Automatic Thread Control in PV-WAVE® 10.0

PV-WAVE Performance Improvements Brochure

by Rogue Wave Software

Printed in the United States of America

Publishing History:

April 2012

TABLE OF CONTENTS

Abstract

During the last several releases of PV-WAVE, there has been a concerted effort to enhance performance. For PV-WAVE versions from 8.0 to 9.0 we parallelized the majority of parallelizable array operations. In PV-WAVE 9.5 we introduced Automatic Thread Control (ATC), an automatically-tuned runtime system that optimizes parallel performance. And for PV-WAVE 10.0, we enhanced ATC, and we parallelized most of the array operations that were still serial, most notably the multi-dimensional operations like subscripting, tiling, and concatenation.

This document describes the advantages of ATC over manual threading control. It also highlights the newly parallelized functionality for this most recent release. For a full listing of OpenMP-enabled functionality refer to the PV-WAVE documentation.

Automatic Thread Control (ATC)

ATC is now available for PV-WAVE on Linux platforms, and on Windows platforms where it was introduced for PV-WAVE in version 9.5, it has been much enhanced.

PV-WAVE uses OpenMP to parallelize nearly all parallelizable array operations, and ATC is an automatically-tuned runtime system which automatically adjusts the number of threads so that it is optimal for any given operation, regardless of array size, data-type, and computing platform.

ATC circumvents a common problem in array-programming: inefficient thread usage. The problem is best illustrated with a few plots.

Figures 1-3 are speedup plots for copying a byte array, for adding a scalar to each element of a byte array, and for computing the hyperbolic-tangent of each element in a DCOMPLEX array. The x-axis indicates array sizes of 2^x elements, the y-axis indicates speedups, the green curve indicates the default method of thread control where a loop is parallel if and only if it exceeds 999 iterations, and the white curve indicates ATC where a loop is run in parallel if and only if it has been predetermined to be faster than running it serially. The plots were generated on a Dell running RHEL6-64 with 2 X5650's each with 6-cores, 12MB cache, and hyper-threading off.

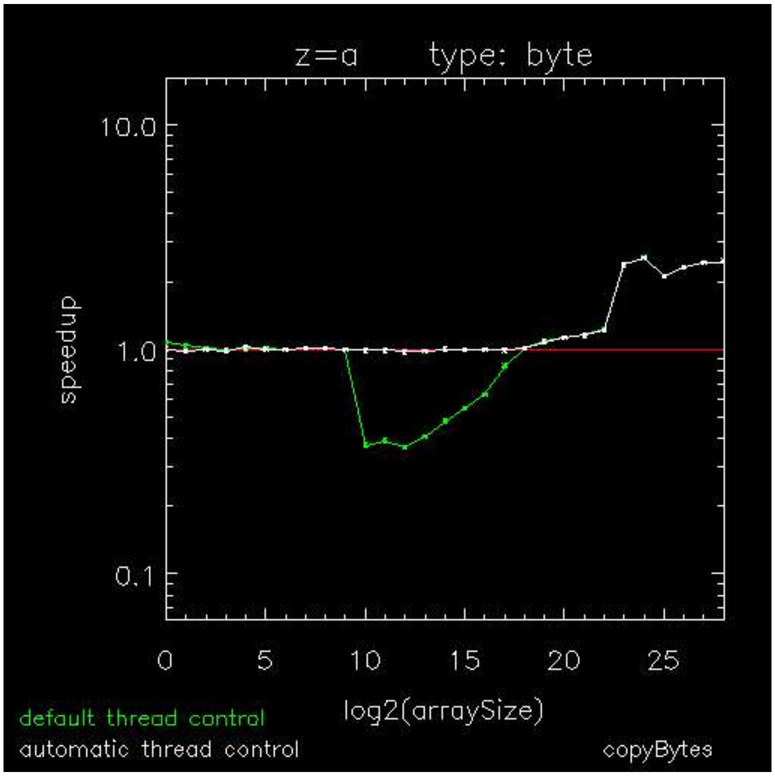

Figure 1 – Copying an Array of BYTEs

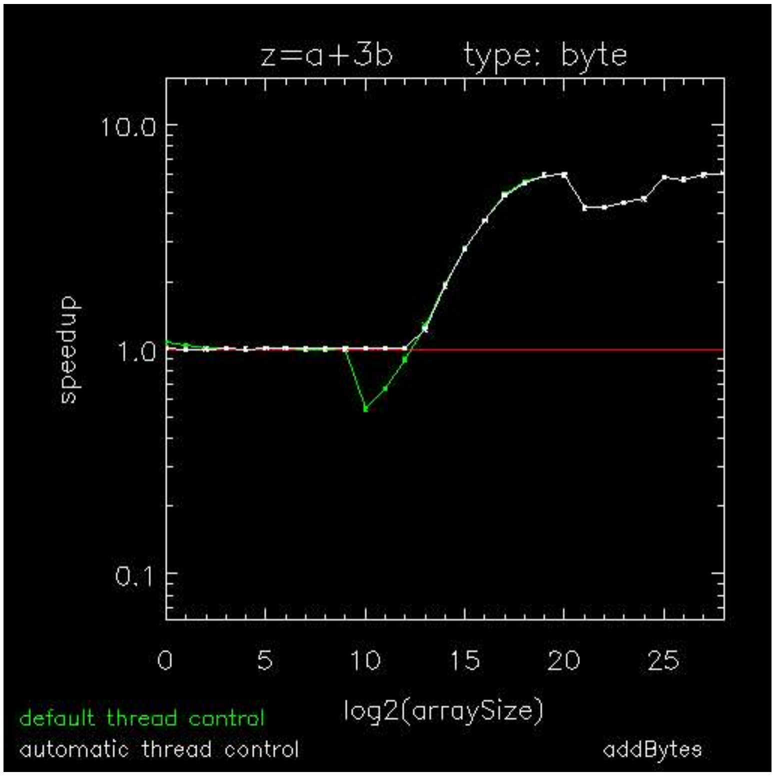

Figure 2 – Addition Using BYTE Values

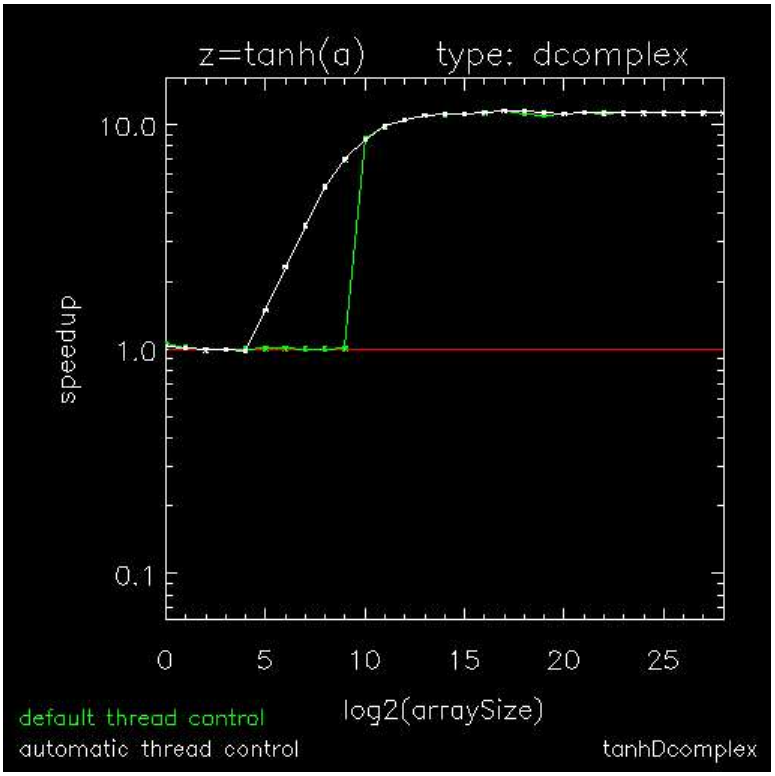

Figure 3 – Hyperbolic Tangent of Double Complex Values

Figures 1 and 3 show extremes for parallelization threshold (the operation size above which parallelization is efficient) and Figure 2 is an intermediate case. On this platform as on most platforms, parallelization threshold can vary by at least four orders of magnitude from one operation to the next, so using a fixed threshold impedes the performance of medium-sized operations with too many threads (Figures 1 and 2) or with not enough threads (Figure 3). Although medium-sized array operations are computationally cheaper than the larger ones, they are just as important to optimize because they can sometimes dominate the work in an unparallelizable loop, itself composed of parallel array operations.

Any attempt to programmatically control the threshold in a large array-based program would be prohibitively time-consuming for the programmer.

Fortunately, the PV-WAVE programmer can launch a simple script that tunes PV-WAVE to any Linux or Windows platform. Once tuned to a particular platform, PV-WAVE automatically adjusts the number of threads at runtime so that it is always optimal for any operation in any PV-WAVE application running on that platform.

The previously discussed Figures 1-3 show how ATC improves the performance of individual element-wise array operations. Figures 4-6 show how it improves the performance of multi-dimensional array operations and composites of array operations where the penalties associated with a fixed parallelization threshold can become compounded.

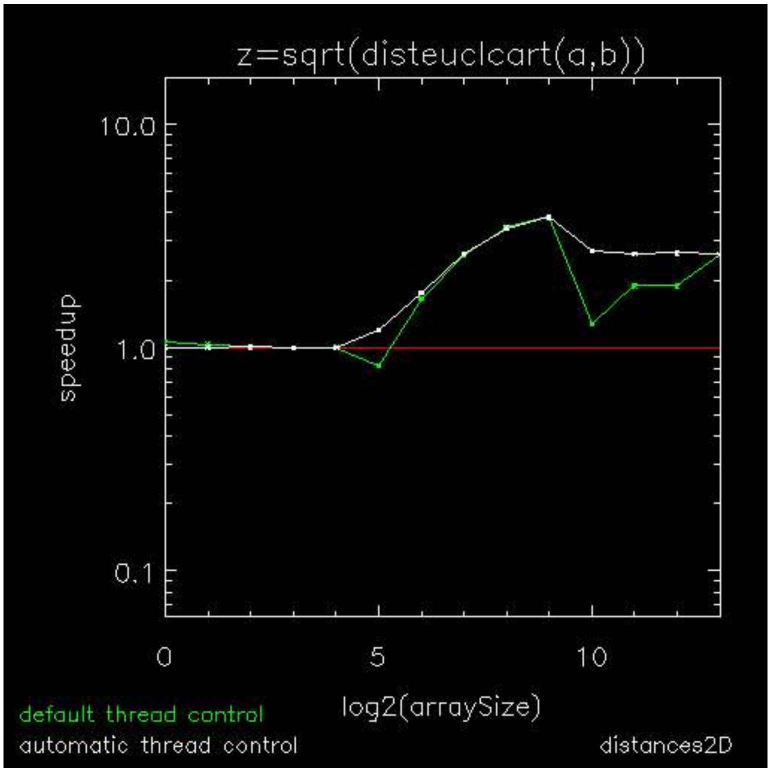

Figure 4 – Composition of Several Array Operations

Figure 4 is a speedup plot for a composite of the several array operations (SQRT and those in disteuclcart.pro) used to compute the matrix of pair-wise distances between two sets of points.

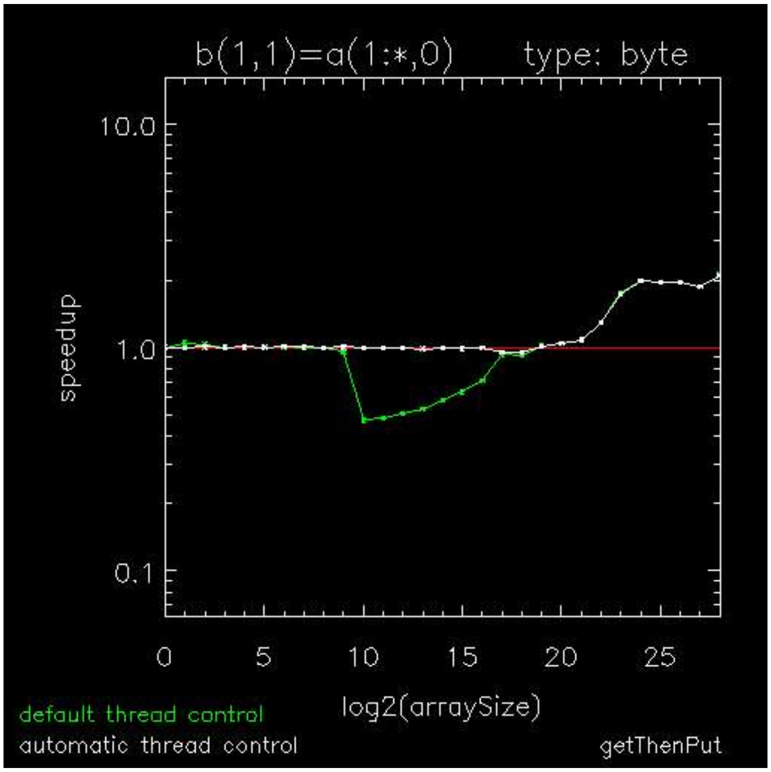

Figure 5 – Sequence of Array Subscripting Operations

Figure 5 shows the performance for a composite of operations used to replace part of a column in a 2d array with part of a column from another 2d array.

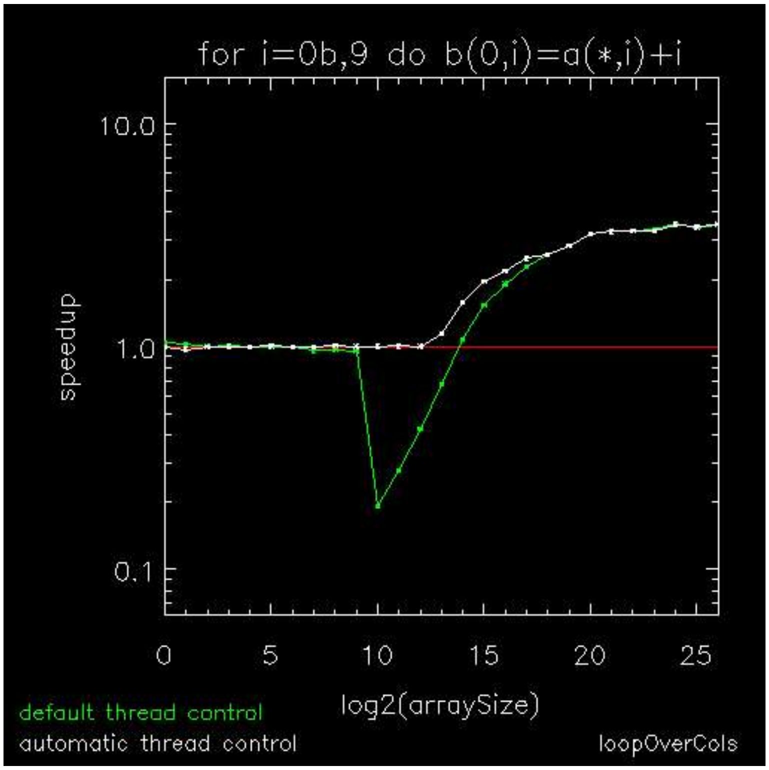

Figure 6 – Looping Over Array Columns with Addition

Figure 6 shows the performance of a sequence of operations similar to those shown in Figure 5, but where each replacement column has a scalar added to it.

The six speedup plots are typical of speedup plots for any operation or composite of operations, where ATC is seen to improve performance by degrees which vary with the operation and where the improvements are always limited to the intermediate sizes of the operation. Regardless of dimensionality, virtually all PV-WAVE array operations are subject to ATC: initializations, copies, reverses, tilings, type conversions, subscripting, searches, intersections, concatenations, transcendental functions, and logical, relational, arithmetic, matrix, and tensor operations.

Newly Parallelized Array Operations

Many previously serial array operations are now parallel and subject to ATC:

N-D Subscripting
N-D Index Conversions (INDEX_CONV)
N-D Concatenation (the [] operators)
N-D Tiling (REBIN with Sample keyword)
N-D Cartesian Products (CPROD)
Order Reversals (REVERSE)
Index Generation (LINSPACE)
Type Conversions (BYTE, FIX, INT32, LONG, FLOAT, DOUBLE, COMPLEX, DCOMPLEX)
Complex Utilities (IMAGINARY, CONJ)
Searches (WHERE)
Greatest, Smallest, and Nearest Integer functions (GREAT_INT, SMALL_INT, NINT)

Parallelization of these array operations produces speedups in computational array-based PV-WAVE code. For example, the RBF function-fitter RBFIMSCL is a large PV-WAVE application which makes extensive use of the new parallel array functionality, and for medium to large problems on multi-core desktop hardware, it now runs more than twice as fast as it did in PV-WAVE version 9.50.

Conclusion

For computationally-intensive array-based processing of medium to large data sets, the newly parallel functionality and Automatic Thread Control in PV-WAVE 10.0 can reduce runtimes by more than a factor of two over PV-WAVE 9.50.

About Rogue Wave Software

Rogue Wave Software, Inc. is the largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications. Rogue Wave marries High Performance Computing with High Productivity Computing to enable developers to harness the power of parallel applications and multicore computing. Rogue Wave products reduce the complexity of prototyping, developing, debugging, and optimizing multi-processor and data-intensive applications. Rogue Wave customers are industry leaders in the Global 2000, ISVs, OEMs, government laboratories and research institutions that leverage computationally-complex and data-intensive applications to enable innovation and outperform competitors. For more information, visit http://www.roguewave.com.