Nova Visualization for Optimization of Data-Parallel Programs *

Clinton L. Jeffery, Sandra G. Dykes, Xiaodong Zhang,
Guillermo H. Gonzalez, and Jason L. Peacock

University of Texas at San Antonio
Division of Computer Science
San Antonio, Texas 78249

Abstract. The nova radial plot in the *Graph system is used to display communication in data-parallel programs. Nova visualizations play a valuable role in understanding how virtual communication in data-parallel programs translates into network communication, and how source code modifications and data layout change physical communication patterns. A case study demonstrates how *Graph nova views are used to visualize and reduce network communication.

Introduction

If communication overhead in a parallel program is to be minimized, the data distribution must not generate unnecessary communication nor cause network hot spots. In message-passing languages the application is responsible for distributing data across processing nodes and for specifying communication details. In data-parallel languages the compilers and run-time systems provide an abstract layer that frees programmers from handling data layout and explicit network communication. Unfortunately, current data-parallel systems cannot anticipate a program's dynamic communication patterns. Without user guidance such as layout directives, system layouts may not match the communication pattern, and may generate unnecessary communication. One of our goals is to visualize the network communication that results from interactions between data layout and virtual processor communication code. This helps programmers improve algorithms and choose a data layout that best fits an algorithm. Although data-parallel languages can be standardized across platforms, the data layout depends directly upon hardware and system configuration. A tool that visualizes data-parallel communication and aids in layout optimization could improve performance on one platform, and retain performance when porting to a new platform. Towards this goal, we have developed *Graph [2], a prototype tool that provides

* This work is supported in part by the National Science Foundation under grants CCR-9102854, CCR-9409082 and CCR-9400719, by the Air Force Office of Scientific Research under grant AFOSR-95-1-0215, and by the Southwestern Bell Foundation. Experiments were conducted on machines in Los Alamos National Laboratory and in the National Center for Supercomputing Applications at the University of Illinois.

event-driven monitoring, visualization and layout optimization for data-parallel communication. This paper describes a new *Graph visualization metaphor, the nova radial plot, and explains its advantages in depicting temporal and spatial distributions in network communication.

Nova Visualization

The nova metaphor (Figure 1, left) is a 2-D radial plot originally developed for the visualization of memory allocation [3]. In this section we apply the visualization principles of Heath, Maloney and Rover from reference [4] to evaluate the effectiveness of nova for visualizing communication.

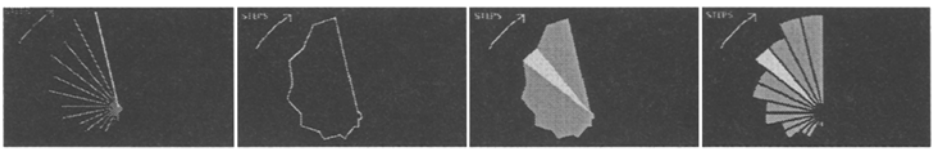

Fig. 1. Nova, nova envelope, Kiviat diagram and filled-arc nova.

In *Graph novas, the line angle θ represents a discrete quantity such as processor number, and line length r represents a continuous variable such as number of message bytes. Color, line width and line style convey additional information. Because of its symmetry, the overall shape of a radial plot efficiently summarizes behavior. At the same time, the individual lines of longest length stand out and attract the user's attention. As a result, the radial characteristic of the nova makes it easy to simultaneously identify overall patterns and to spot outstanding individual points. In cases with a large number of values to be plotted, details can be retained by animation; the nova animates naturally with a radar-sweep motion that avoids the discontinuity of cartesian graphs when they "wrap around" the edges.

Radial geometry is used in other visualization techniques such as kaleidoscopes [5] and Kiviat diagrams. Unlike the detail-oriented and lightweight nova, kaleidoscope visualizations typically summarize entire program execution in a single **signature** or snapshot. Kiviat diagrams are similar to the envelope formed by connecting nova endpoints, although Kiviat sections are normally shaded (Figure 1). Choice of the most appropriate technique depends on the relationship between adjacent points: novas are appropriate when adjacent points are independent, while Kiviat diagrams are appropriate when adjacent points are related. However, Kiviats can be misleading if used to compare data sets of unequal size. This is because Kiviats draw attention to the area between adjacent points, and that area depends upon number of points as well as point values. Conversely, nova radial lines are plotted independently, which keeps attention focused on individual points rather than on changes between points. Consequently,

we have found nova to be the most appropriate metaphor for depicting communication; it presents information without implying relationships that may not exist between communication events or processor nodes that are adjacent in the visualization.

Another interesting case arises when every other data point in the plot has a zero value. This occurs, for example, when alternating processors send a message and their adjacent neighbors receive a message. Here the nova and Kiviat diagrams appear identical and share the problem that n element plots with alternating zero values are indistinguishable from $n/2$ plots with no zero values. Unlike Kiviat diagrams, the nova plot can remove this ambiguity by scaling line width according to number of data points, thereby producing a *filled-arc nova* (Figure 2).

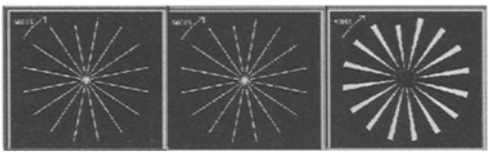

Fig. 2. Nova, Kiviat diagram and filled-arc nova for *sends* in a data-parallel program where alternating processors send and receive messages. Every other radial line has zero value.

Nova Views of Temporal and Spatial Communication

*Graph's main window (Figure 3) consists of 1) temporal novas, 2) spatial novas and 3) a source code listing. **Temporal novas** show when communication occurs during the execution of a program: r = *message bytes*, θ = *communication step* . In the data-parallel paradigm, a communication step refers to the synchronous execution of a source code communication instruction. Therefore each line in a temporal nova corresponds directly to a source code statement. Although each nova line refers to a single code statement, one statement may be executed multiple times, and may generate many nova lines. When the user selects a temporal nova line, *Graph highlights the corresponding source code statement and displays the spatial novas for the selected step. To find source code statements which generate communication bottlenecks, the user selects the longest nova lines.

Spatial novas show where (on which processing nodes) communication occurs for a selected step: r = *message bytes*, θ = *processor node*. As with the source code display, the spatial novas change when a new step is selected. This enables users to detect if a traffic "hot spot" exists for a given step. The nova makes it easy to identify several common communication patterns. For example, full circle patterns in both the send and receive novas indicate *one-to-one* communication, whereas a full circle in the send nova and a single line in the receive nova indicate *all-to-one* communication.

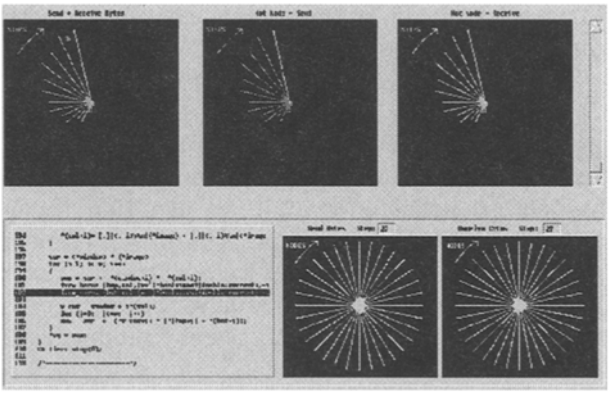

Fig. 3. *Graph main window for the folded convolution with default layout, showing temporal novas (top), spatial novas (bottom right) and source code listing (bottom left). The three temporal nova display total message bytes (left), send bytes (middle) and receive bytes (right).

Data-parallel convolution: a case study

This study examines two data-parallel convolution algorithms implemented in C* on the CM-5 using 32 nodes, with image size 256×256 and mask size 15×15. The first is a straightforward implementation in which each virtual processor sends to all neighbors within the mask. Communication occurs inside a doubly nested (row,column) loop, generating the maple leaf pattern in Figure 4 (right). A better algorithm is the folded convolution [1] in which virtual processors send single elements to row neighbors, then send arrays to column neighbors. *Graph visualizations for the convolution algorithms are shown in Figures 3 and 4.

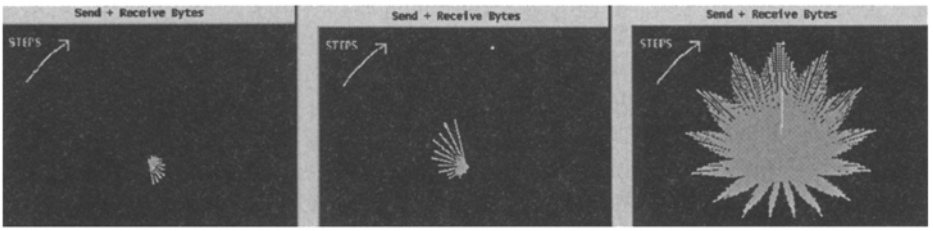

Fig. 4. Scaled temporal novas for folded convolution program with optimized layout (left), folded convolution program with default layout (middle), and naive convolution program (right).

*Graph determined the compiler's default block checkerboard layout is optimal for the naive case, but suggested a block striped layout for the folded case. This stores more column elements on the same node, reducing remote messaging for column accesses. Temporal novas for the default layout show most communication occurs during the last few steps. By selecting the longest nova line, we

see highlighted the source code instruction for array *sends*. After layout optimization, temporal novas lines appear only for the first half of execution, which correspond to single element *sends*.

Scaled temporal novas in Figure 4 graphically illustrate how communication is reduced by improving the algorithm and by optimizing data layout. Table 1 contains message bytes, execution and communication times. Although improving the algorithm had an even larger effect, layout optimization was significant as it reduced messages from 7.8 MB to 3.7 MB, and execution time by 31%.

Algorithm	Data Layout	Exec. Time	Comm. Time	Local data	Remote data
Naive	default = opt.	0.38 s	0.34 s	98.6 MB	19.9 MB
Folded	default	0.13 s	0.10 s	58.3 MB	7.8 MB
Folded	opt.	0.09 s	0.06 s	62.4 MB	3.7 MB

Table 1. Execution and point-to-point communication times for the case study.

Summary

The straightforward statistical displays used in most other parallel visualization projects are insufficient for complex data-parallel behavior and performance. Nova radial plots make it easy to simultaneously discern overall temporal and spatial communication patterns, and to identify points of large communication. By relating nova lines to the source code listing, *Graph helps users understand how algorithms, implementations and data layouts affect communication. The nova visualization addresses several important performance issues, including system scaling, execution pattern comparisons and extraction of performance data. Our current work includes investigating use of *Graph on networks of workstations for data-parallel programs. In addition to visualizing load balancing and effects of heterogeneity, we plan to animate communication on switch networks, such as ATM and Myrinet. This will show where network communication occurs and how that locality changes over time.

References

1. S. G. Dykes and X. Zhang, "Folding spatial image filters on the CM-5", **Proc. of the 9th IPPS** IEEE Computer Society Press, April, 1995, pp. 451-456.
2. S. G. Dykes, X. Zhang, Y. Shen, C. L. Jeffery and D. W. Dean, "*Graph: a tool for visualizing communication and optimizing memory layout in data-parallel programs", **Proc. of 1995 ICPP**, CRC Press, Vol. II, August, 1995, pp. 121-129.
3. R. E. Griswold and C. L. Jeffery, "Nova: low cost data animation using a radar sweep metaphor", **Proc. of UIST'94**, November, 1994, pp. 131-132.
4. M. T. Heath, A. D. Malony and D. T. Rover, "The Visual display of parallel performance data", **IEEE Computer**, Nov., 1995, pp. 21-28.
5. E. Tick and D.-Y. Park, "Kaleidescope Visualization of Fine-Grain Parallel Programs", **Proc. of the 25th Hawaii International Conference on System Sciences**, vol. II, 1992, pp. 137-148.