The NAS Trace Visualizer (NTV) Rel. 1.2 User's Guide[1]

Louis Lopez[2]

Report NAS-95-018 September 95

NAS Systems Division
NASA Ames Research Center
Mail Stop T27-A
Moffett Field, CA 94035-1000

Abstract

One of the more useful techniques for debugging and tuning parallel programs is the tracing of program execution. Because the volume of data produced can be overwhelming, and because the data is usually in a cryptic form as a result of trying to minimize the perturbations of the execution caused by collecting the data, effective use of this technique requires a tool to visualize the trace. This document describes **NTV**, a trace visualization tool developed at NAS. **NTV** differs from most visualization tools in that it uses displays that do not vary with time, *i.e.*, static displays, rather than the animated displays that are common in other visualizers. Zooming is supported as are other display functions such as panning and controlling the display of message edges.

1. A full color Postscript version of this report is available from the NTV Web page: http://www.nas.nasa.gov/NAS/TechReports/
2. Computer Sciences Corporation, NASA Contract NAS 2-12961, Moffett Field, CA 94035-1000

1.0 Introduction

NTV is a Motif-based trace visualization tool for use with trace files produced using *The Automated Instrumentation and Monitoring System (AIMS) (Version 2.2 or 3)* or the *IBM SP2 MPL Trace Facility*. Unlike most trace visualizers, where animated displays are common, **NTV** uses displays that do not vary with time.

NTV provides a *time line* display and a set of *profiling* displays. The *time line* display presents a detailed view of the program execution showing the state of the processors at every point in time and showing the message traffic between processors. This is done using a color-coded time line for each processor. The color coding shows the processor status. Message edges are shown by lines drawn between processor time lines. Various types of zooming and panning are supported, as is the ability to control which message edges are shown.

The profiling displays present summary information in which data for the execution is summarized in two ways. The *processor profiles* allocates information by processor, *e.g.*, the time the processor was blocked while sending, and the *function profiles* allocates information by function, *e.g.*, the time a function was blocked while sending.

NTV detects the trace type so the same version of **NTV** works for all the supported trace formats. While the operation of **NTV** is essentially the same for all the supported traces, there are some differences in the information presented because the information in the different traces is not identical. Where differences do exist in the operation of **NTV**, they will be described. If some characteristic is not specifically flagged as not applying to all the traces, it applies to all.

2.0 Time Line Display

The *time line* display (Figure 1) presents a set of horizontal bars, one for each processor. The bars are color coded to show the status of each node. The horizontal axis is time in seconds.The meaning of the colors is presented at the bottom of the display. The axis is annotated with the minimum displayed time at the left, and the maximum displayed time at the right. The time displayed at the center of the X-axis is the time interval between the vertical lines that bound the region selected for zooming (the vertical lines extending above Processor 0 in Figure 1). The zoom range in Figure 1 is 0.00453 seconds. If no zoom region is selected, the time interval displayed is the time between the left and right boundary of the display.

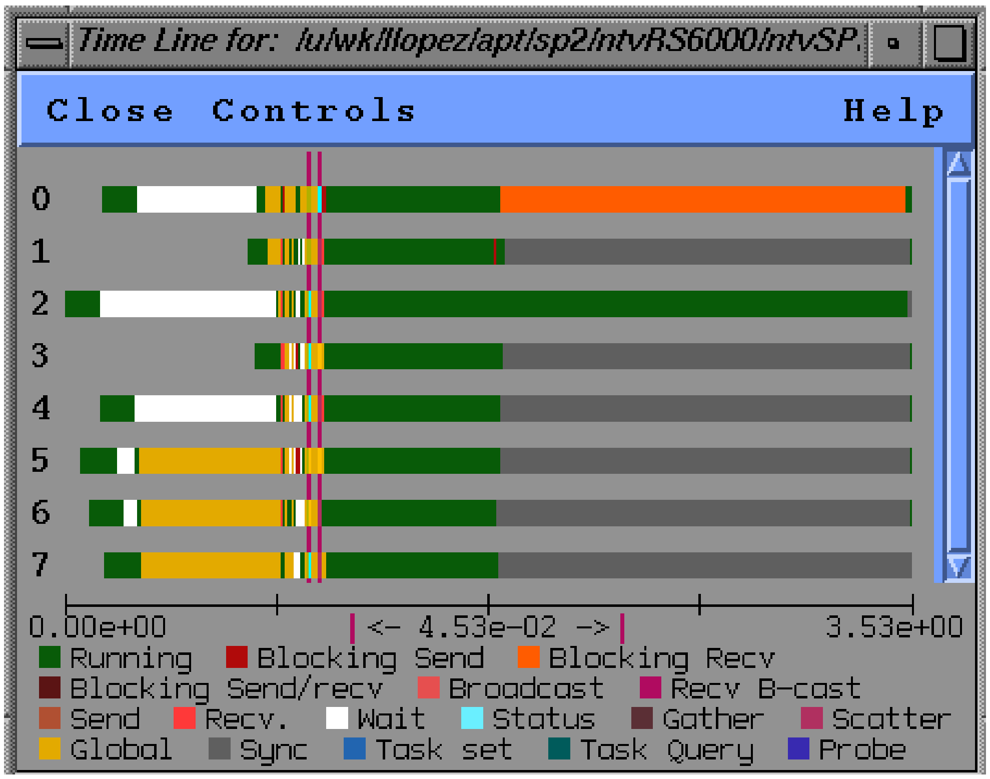

Time Line for: /u/wk/llopez/apt/sp2/intvRS6000/intvSP.
Close Controls Help
0
1
2
3
4
5
6
7
0.00e+00
<- 4.53e-02 ->
3.53e+00
🟩 Running 🟥 Blocking Send 🟧 Blocking Recv
🟫 Blocking Send/recv 🟥 Broadcast 🟪 Recv B-cast
🟫 Send 🟥 Recv. ⬜ Wait 🟦 Status 🟫 Gather 🟪 Scatter
🟨 Global ⬛ Sync 🟦 Task set 🟩 Task Query 🟦 Probe

Figure 1.

The display can be resized in the vertical direction to show more processors, or the scroll bar can be used to scroll the display.The top of the display is a menu bar containing a button to close the display, a button to open a *Control panel* and a help button. A variety of controls are possible using the mouse and the *Control panel* described in Section 2.1.

One use of the *Control Panel* is to control the display of message edges. Figure 2 shows the display in Figure 1 with all message edges displayed. The solid vertical block in Figure 2 is caused by message edges overlapping because a large number of message edges are drawn in a small area. This area can be expanded by zooming as explained below.

A significant amount of navigation functionality is available in the mouse buttons. The left mouse button is used to select a region for zooming or positioning. To select a region, the cursor is placed at the starting point of the region and the left mouse button pressed. A vertical line will appear indicating the start of the region. Without releasing the button, the mouse is dragged until the cursor is at the end of the region and the button released. A second vertical bar will indicate the end of the region. Pressing the middle button will zoom into this region. Figures 1 and 2 show a selected zoom region of 0.0453 seconds. Figure 3 shows the display in Figure 2 after zooming into an even smaller region (0.00325 Seconds of the execution).

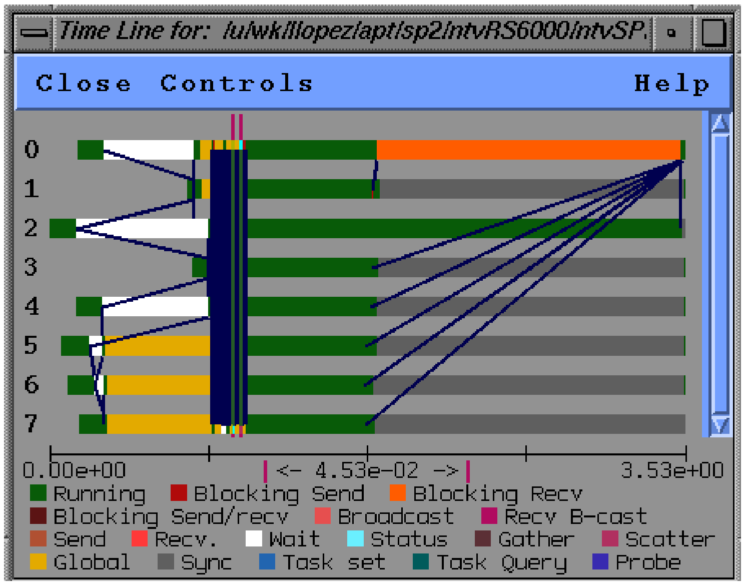


Figure 2.

If the zoom region is a single line, selected by pressing and releasing the left mouse button without dragging the mouse, then pressing the middle mouse button will cause the display to be centered about the selected line. (See the discussion of the control panel in the next section).

A source code browser, described later, can be opened on the statement that caused the trace record to be generated by placing the cursor on a time line and pressing the right mouse button.

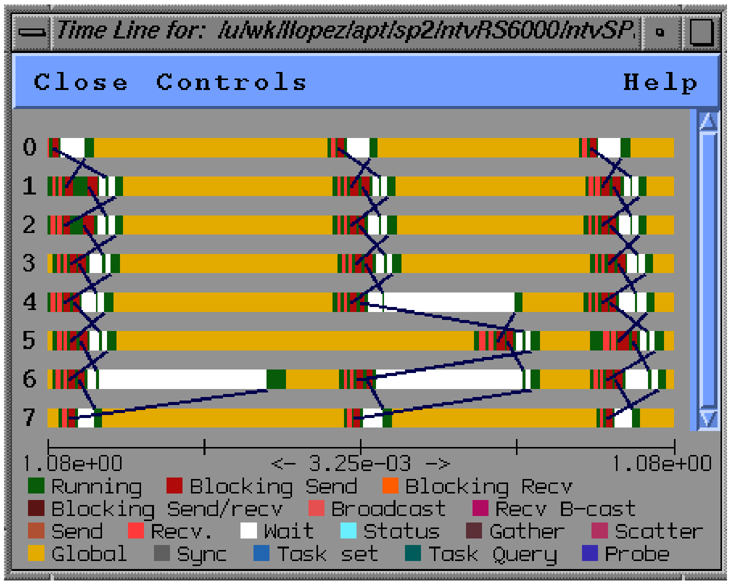
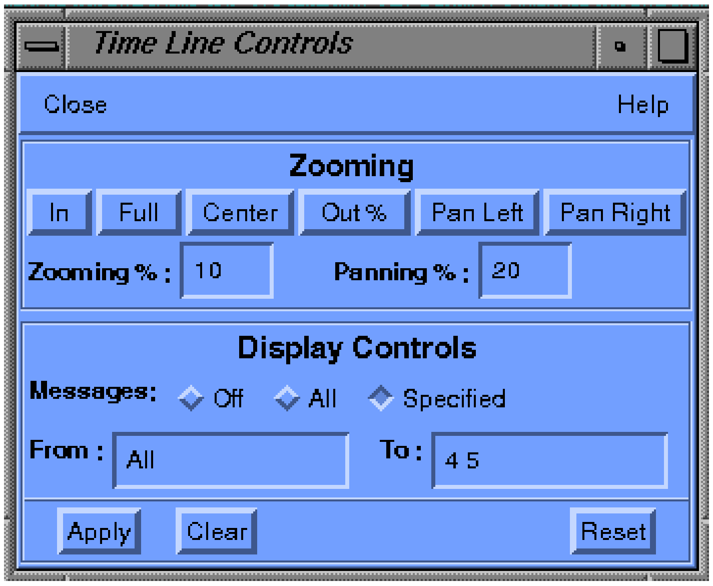


Figure 3.



Zooming % : 10

Panning % : 20

Messages: Off All Specified

From : All

To : 4 5

Figure 4.

2.1 Time Line Control Panel

The *time line control panel* (Figure 4) is opened by selecting the *Controls* button in the *time line* display. The control panel lets you control various aspects of the *time line* display. It has two sections, one related to positioning of the display and the other to the display of message passing edges.

2.1.1 Positioning Controls

In and Center Buttons

The “In” and “Center” buttons operate identically to the use of the middle mouse button. Note that zooming in, or out, causes the data to be rescaled while centering just moves the display window keeping the same scaling. Because centering does not re-scale, it may be impossible to center, *i.e.*, if the centering point is too close to either end of the trace data set. For these cases the display will be centered as much as possible, *i.e.*, the display will either began at the start of the run, or it will end with the end of the run.

Full Button

The “Full” button will cause the display to show the complete trace file.

Out %

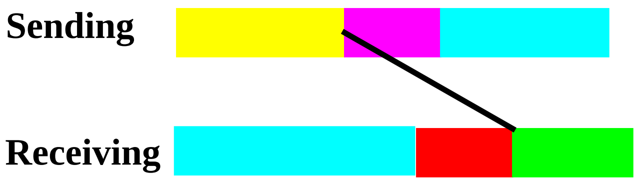
The “Out %” button zooms out by the percentage in the “Zooming %” text window. This percentage is set to 10% but can be reset by modifying the number in the “Zooming %” text window. The percentage is based on the current display so that if 1 second of a trace is being shown and the zooming percent is 10%, the new display will show 1.1 seconds of the trace and if 25 seconds is being shown the new display will show 27.5 seconds of the display. If possible, the display will be centered at the same point as the original display.

Pan Buttons

The “Pan” buttons operate much as would a horizontal scroll bar. They move the display window over the data in the pan direction, *e.g.*, if you pan left you see data to the left of the current window. The “Pan” buttons are used instead of a scroll bar because for large trace files it is difficult to respond quickly enough to support smooth scrolling. The value in “Panning %” is the amount the display is moved. When you pan, a line will show the position of the old display boundary, *e.g.*, if you pan left by 20% a vertical line will appear 20% of the way from the left edge.

2.1.2 Message Display Controls

Message edges are displayed as blue lines between the processors involved in the message exchange. The message edges begin at the center line of the sending time line and end at the edge of the receiving time line, as shown below. The default is to not show the messages.



Radio Buttons

The “Off” button turns off the display of message edges (the default).

The “All” button turns on the display of all message edges.

The “Specified” button turns on the display of only those message specified in the “From” and “To” lists. The from and to lists accept sets of integers indicating the nodes between which message edges are to be displayed, *e.g.*, if “From” contains 1 3 and “To” contains 2

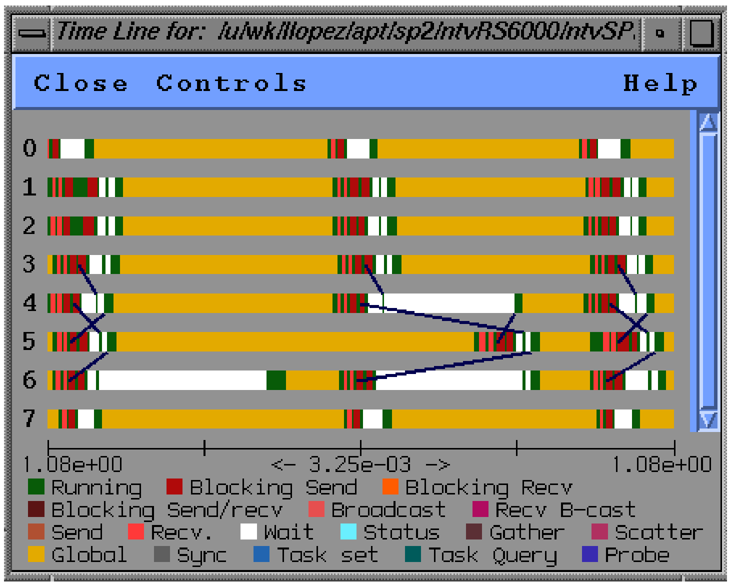


Figure 5.

4, all message from 1 to 2 and 4, and from 3 to 2 and 4 will be shown. The word “All” or “all”[1] can be used to indicate all nodes.

Apply Button

The changes in the “Display Controls” section do not take effect until the “Apply” button is pressed.

The “Display Controls” section of Figure 4 shows a selection to display only specified message edges. The selection is message edges from all processors to processors 4 & 5. If this selection is applied to the display in Figure 3, one gets the display in Figure 5.

3.0 Summary Displays

The *Summary Displays* present profiling information. They are of two types. One assigns data to functions *(Function Profiles)* and the other to processors (*Processor Profiles).*

Function Profiles

The *Function Profile* displays (Figure 6) differ in the data they present, but use the same method of presentation. The display consists of a column of names of functions and user defined blocks. Each name is followed by a bar chart, followed by a number indicating the number of processors on

1. **NTV** only checks the first character to see if it is ‘A’ or ‘a’.

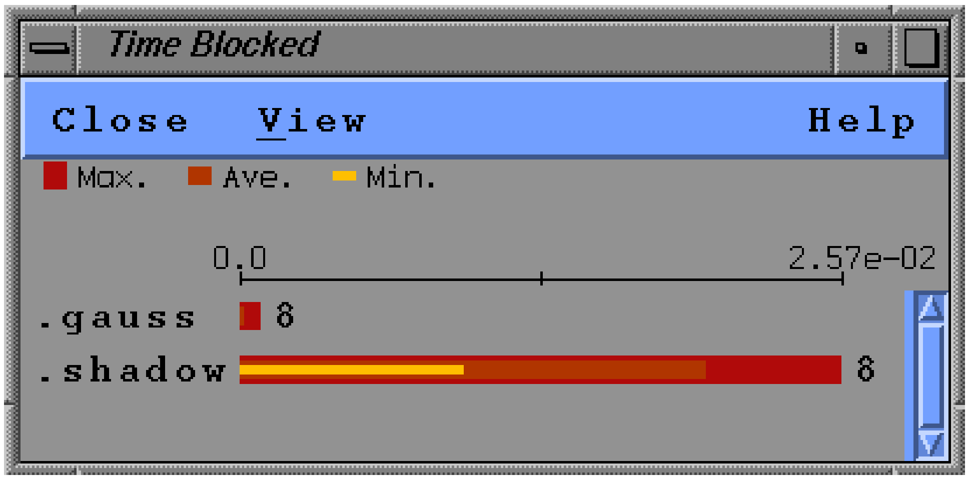


Figure 6.

which the function or block ran. *e.g.*, Figure 6 shows that functions *gauss* and *shadow* each ran on 8 processors.

For *AIMS* traces, names of code blocks, e.g., do loops, are preceded by one or more asterisks (*). The blocks are displayed following the function of which they are a part. The number of asterisks indicate the level of nesting, *e.g.*, two asterisks indicate a block within a block.

The bar chart used for each function (See Figure 6) consists of 3 overlaid bars. The widest bar is the maximum data value recorded on any processor, the narrowest is the minimum recorded on any processor, and the middle width bar is the average over all processors on which the function or block executed.The displays presents the data for every function in the program, so if some block is not instrumented it will show zero time for those data items not instrumented. If the function or block executed on only one processor the "Max", "Min.", and "Avg." are all the same so the bars will look like a series of parallel lines.

The display is scaled to the longest displayed time. The display can be made larger or smaller using the usual window manager resizing operations. The larger the vertical dimension, the greater the number of functions that will be displayed. The scroll bar can be used to scroll the display.

Clicking on any bar with the middle mouse button will open a *detail* display. This display gives the processor number and the numerical values of the minimum and maximum used in the display.

Clicking on any bar with the right mouse button will open a browser showing the code for the function or user defined block corresponding to the bar. One browser can be opened for each display. If a browser is already open and the right button is used to click on another bar, the code corresponding to the new selection will be shown.

The menu bar at the top of the window has 3 buttons. The “Close” button is used to close the display, and the “Help” button to get explanatory material on the specific display. The “View” button is used to bring down a menu to control the operation of the left mouse button. The default is that the left button does nothing. If the “Details” option is selected, the left mouse button operates identically to the middle mouse button, *i.e.*, it opens and controls the information in the *Detail* display. If the “Browser” option is selected, it operates identically to the right button, *i.e.*, it controls the Browser.

There are 6 *Function Profile* displays for *AIMS 2.2* traces:

Execution	The time spent executing (*i.e.*, not blocked, idle or flushing).
Send-Sync Blocked	The total time spent blocked on synchronized sends.
Send-Async blocked	The total time spent blocked on asynchronous sends.
Receive-Sync Blocked	The total time spent blocked on synchronous receives.
Receive-Async Blocked	The total time spent blocked on asynchronous receives.
Global Blocked	The total time spent blocked for global operations.

There are 14 *Function Profiles* for *Aims 3* traces, the 6 above plus the following 8 additional ones:

Write (Time)	The total time spent writing.
Write (Vol.)	The total volume (bytes) written.
Read (time)	The total time spent reading.
Read (Vol.)	The total volume (bytes) read.
Pack (time)	The total time spent in pack operations.

Pack (Vol)	The volume of data packed (bytes).
Unpack (Time)	The total time spent in unpack operations.
Unpack (Vol)	The total volume of data unpacked (bytes).

There are 10 *Function Profile* displays for *SP2* traces:

Send Volume (Un-Blocked)	The volume of messages sent using non-blocking sends.
Send Count (Un-Blocked)	The number of messages sent using non-blocking sends.
Send Volume (Blocked)	The volume of messages sent using blocking sends.
Send Count (Blocked)	The number of messages sent using blocking sends.
Recv. Volume (Un-Blocked)	The volume of data received (in bytes) using non-blocking receives.
Recv. Count (Un-Blocked)	The number of messages received using non-blocking receives.
Recv. Volume (Blocked)	The volume of data received (in bytes) using blocking receives.
Recv. Count (Blocked)	The number of messages received using blocking receives.
Wait State	The time spent in wait states.
Time Blocked	The time spent in blocked states.

3.1 Processor Profiles

The *Processor Profile* displays consist of sets of stacked bars for each processor (see Figure 7). The bars are color coded to indicate the data they represent. The legend at the top of the display shows the color coding used.

The display is scaled to the longest displayed time. The display can be made larger or smaller using the usual window manager resizing operations. The larger the vertical dimension, the greater the number of processors that can be displayed.The scroll bar can be used to scroll the display.

Clicking on the bars for any processor with the middle mouse button will open a *detail* display. This display shows the processor numbers and the actual data used to create the display.

The menu bar at the top of the window has 3 buttons. The "Close" button is used to close the display, and the "Help" button to get explanatory material on the specific display. The "View" button is used to bring down a menu to control the operation of the left mouse button. The default is that the left button does nothing. If the "Details" option is selected, the left mouse button operates identically to the middle mouse button, *i.e.*, it opens and controls the information in the *Detail* display.

The browser can not be accessed from this display.

There are 5 *Processor Profiles* for the *AIMS 2.2* traces:

Totals (Time)	The time running, the time blocked in communications, the time blocked for global operations, and the total time blocked for all reasons.
Blocked (Time)	The time blocked for sends, for receives, for global operations, and the total time blocked.
Totals (Volume)	The volume of data, in bytes, sent and the volume received.
Send (Volume)	The volume of data, in bytes, sent in each of the modes synchronous, synchronous blocked, asynchronous, and asynchronous blocked.
Receive (Volume)	The volume of data, in bytes, sent in each of the

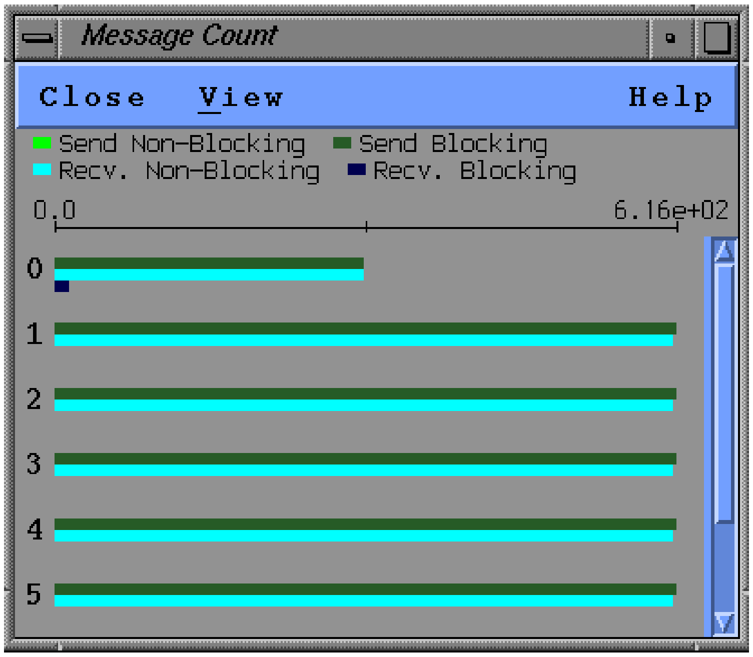


Figure 7.

	modes synchronous, synchronous blocked, asynchronous, and asynchronous blocked.

There are 9 *Processor Profiles* for *Aims 3* traces, the 6 above plus 4 others:

I/O (Time)	The time spent in read and in write operations.
I/O (Vol)	The volume of data (bytes) read and written.
Pack/Unpack (time)	The time spent in pack and in unpack operations.
Pack/Unpack (Vol)	The volume of data (bytes) packed and unpacked.

There are 3 *Processor Profiles* for the *SP2* traces*:*

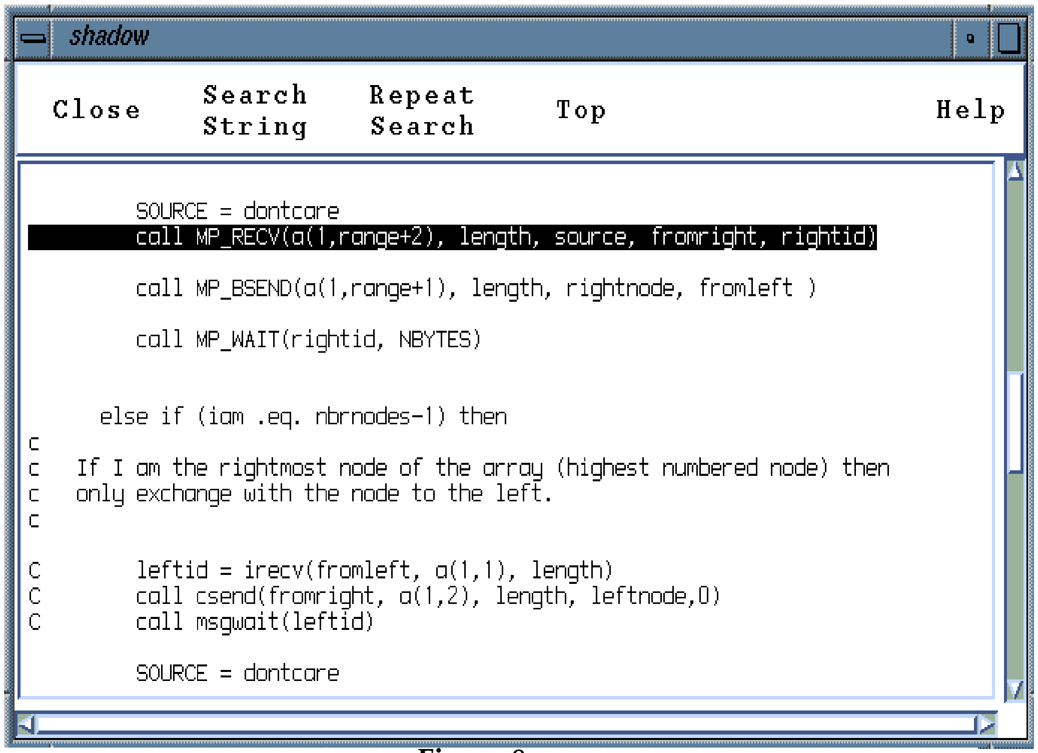
Blocked Time	The time in wait states, the total blocked time, the time blocked on receives and the time blocked on sends.
Message Volume	The volume sent in blocked mode and in non-blocked mode and the volume received in blocked and in non-blocked mode.
Message Count	The number of messages sent and the number received in blocked and in non-blocked mode.

4.0 Source Code Browser

A browser (Figure 8) can be opened from either a *function profile* or the *time line* display. On the *function profile* a browser can be opened by clicking with the right mouse button on any function bar. A browser will open positioned on the code for the function corresponding to the bar.

On the *time line* display the browser is opened by placing the mouse cursor on a processor bar at the point of interest and pressing the right mouse button. This will open the browser positioned at the source code line that caused the generation of the trace record that has the time stamp closest to the time selected on the display.[1] Note that data may be compressed when displayed, *i.e.*, many trace points may scale to one display point when plotted. Because of this, the browser may not be positioned at the point you expect. This problem diminishes as you zoom in. Another

1. Because of limitation in program symbol tables, NTV will not position IBM SP2 programs at the correct line if the program uses *include* files that contain executable code.



```
        SOURCE = dontcare
        call MP_RECV(a(1,range+2), length, source, fromright, rightid)

        call MP_BSEND(a(1,range+1), length, rightnode, fromleft )

        call MP_WAIT(rightid, NBYTES)


      else if (iam .eq. nbrnodes-1) then
c
c   If I am the rightmost node of the array (highest numbered node) then
c   only exchange with the node to the left.
c

C       leftid = irecv(fromleft, a(1,1), length)
C       call csend(fromright, a(1,2), length, leftnode,0)
C       call msgwait(leftid)

        SOURCE = dontcare
```

Figure 8.

possible source of confusion arises from the way traces are generated and the way the displays are generated from the trace.

Trace records are generated to indicate the beginning and end of execution and the beginning and end of communication library events. For example, if a blocking send occurs, a trace record is generated to indicate the start of the blocking send and another record is generated to indicate the end of the blocking send. When execution begins the processor is colored green. When the processor enters a new state the color changes to indicate the new state, *e.g.*, blocking send color. When this state ends the color returns to the previous states color, *i.e.*, to the execution state color. There is no record generated to indicate a return to the run state. This means that the search for the trace record closest to the selected point will usually find a call to the message passing library even if you select a point in the middle of a long run section.[1]

The browser can be repositioned in 4 ways:

1. Using the scroll bar.

2. Making a new selection in the display that opened the browser. The selection is made in the same way as the selection which opened the browser. If the browser function is selected in a window and the browser

1. The possible exception is AIMS traces if entry to functions has been instrumented.

for that window is already open, the code is repositioned. If the requested point is in a file other than the one that is loaded, the new file will be loaded in the same browser window.

In the case of the *function profile* displays, the browser will be positioned at the beginning of the function or user defined block corresponding to the selected bar.

In the case of the *time line* display, the browser will be positioned at the line of code which generated the trace record closest to the point selected and the line of code will be hi-lighted. Note that several trace points may scale to the same display point, so the selection may not be unique. This is particularly true when the display is zoomed out.

3. A search facility that allows entry of a string to be found. The search will wrap if the string is not found between the current position and the end of the file. The search can be repeated by clicking on the "Repeat Search" button. When a new file is read the search string is discarded so the repeat search will not work on the new file. However, if the search window is opened, the last search string will still be displayed in the input area and clicking on "OK" will cause the search. The found string will be hi-lighted.

4. The "Top" button which repositions the file to the top.

You may mark a point by using the mouse and dragging. This type of hi-lighting will be ignored by the browser and is discarded if a new file is loaded. To remove it without loading a new file, click on the marked portion.

4.1 Help Function

All of the **NTV** windows have a help button that gives specific information about the data presented in that display, or information on using the function provided by the display.

4.2 Use of the Mouse

The use of the mouse on each of the displays was discussed before. Their use is easy to remember if it is realized that the mouse buttons have a similar function in all displays. The left button is used to select a point or item, the middle button causes some action based on the selection, and the right button brings up a browser where appropriate. The table below gives the detail.

	Left[1]	Middle	Right
Time Line	Select Range/Point	Zoom/Center	Open Browse
Processor Profile	Select Processor	Show Detail	No Action
Function Profile	Select Function	Show Detail	Open Browser

The Detail Displays present the actual values used to draw the selected bars (See Below). There can be one "detail" window for each display. If you click on a bar in a display which already has an open "detail" window, a new window will not be opened, but the data will change.

When the right mouse button is clicked on a bar in a *Function Profile* display a browser will be opened on the source code for the function or user defined block corresponding to the bar. When the right mouse button is clicked on a time line on the *time line* display, a browser will be opened on the source code that caused the generation of the trace record closest to the selected time on the selected processor. The actual line of code will be highlighted. Note that because of scaling in drawing the display, several trace records may map to one display point so the point selected does not necessarily map to a unique trace record.

One browser can be opened per display. Selecting a different point and pressing the right button will display the new code. The browser has a help facility which describes its operation.

Because so many displays can be open at any one time and sets of the displays are related, the title of every display will contain the name of the principal display, *e.g.*, if you select the *Execution* display, "Execution" will appear in the title of the detail, help, and browser displays opened from the *Execution* display.

5.0 Trace Formats

While the operation of **NTV** is essentially the same for all the supported trace formats, there are some differences. These are described below.

1. The action of the left mouse button in the profiling displays can be altered using buttons in the display menu bar. This is explained in the section discussing the displays.

5.0.1 AIMS

Because a user of **NTV** with *AIMS* traces must have first instrumented the application source code using *AIMS*, it is assumed that the user has read the appropriate *AIMS* documentation. However, some brief description of *AIMS* is necessary to explain **NTV**.

After a program has been instrumented using *AIMS,* there are two versions of the source code, the instrumented version and the un-instrumented version. The instrumented source code is not used by **NTV**, but the un-instrumented source code is used by the source code browser that ties **NTV** displays to the source code. The browser looks for the source code in one of two locations. The first is the directory that contained the source code when it was instrumented[1], and the other is the directory containing the sorted trace file.[2]

AIMS lets you instrument selectively. That is, you can instrument some sections of code and some constructs and not others. If you decide to instrument only some of the code or constructs, some of the displays will appear to give wrong information. For example, if entries to some subroutines are instrumented and the entries to some are not, the execution time will not be assigned correctly in the Function Summary Displays. *AIMS* lets the user define blocks of code that are treated as units. From the standpoint of monitoring these are equivalent to functions, *i.e.*, you can time entry and exit to the block and allocate information such as time spent in the block to the block. When discussing an *AIMS* trace, the term function or subroutine will include user-defined blocks.

5.0.2 SP2

The *SP2* trace facility is described in IBM *AIX Parallel Environment, Operation and Use (SH26-7230)*. To use **NTV** the user's program should be compiled with the `-g` option, to support the source code browser function, and executed with `-tlevel 3`. While `-tlevel 9` can be used, the AIX Kernel Statistics Data is not used by **NTV** and specifying the higher trace level results in a larger file. If the IBM-supplied visualization system is used along with **NTV**, it may be desirable to run the application with `-tlevel 9`.

The *SP2* trace does not mark entry and exit from subroutines, and as a result, it is not possible to allocate run time to individual functions.

1. The *AIMS* trace file contains this directory in the second line of the file.
2. The trace file produced using *AIMS* Version 2.2 must be sorted using the *tracesort* function provided by *AIMS* before it can be used by **NTV**. The sorted trace file can be given any name.

NTV for the *SP2* uses an *SDDF (Pablo*) version of the native *SP2* trace. To get this form of the trace, the native trace must be preprocessed by the program **ntvSP2preProc**. In addition to the native trace, this program also takes as input the *SP2* executable that generated the trace. It produces the *SDDF* file and another file having the same name as the *SDDF* trace file with *".aux"* appended. This file contains information necessary to support the browser function described in Section 4 and to speed up loading of the trace file into **NTV.**

To preprocess the trace file you execute **ntvSP2preProc** and it prompts for three data elements:

- The name of the executable that generated the trace.
- The name of the *SP2* trace file.
- The form of the output file, ASCII or Binary. The Binary file is smaller and will load faster into **NTV**.

ntvSP2preProc prints a dot for each 1000 trace records that are processed. This gives a visual clue that the program is running.

The browser looks for source code in three locations. The first is the directory containing the trace file, the second is a directory named *"src"* at the same level as the trace file, and the third is the directory above the one containing the trace file.

5.1 Allocation of Data in Summary Displays

To understand the Summary Displays you need to understand a little about how **NTV** uses the trace data to generate the profiling displays.

NTV attempts to allocate all data to a function, *e.g.,*the volume of messages sent by each function or time blocked by each function. In the case of *AIMS,* which allows selective instrumentation, the information may not be assigned to the correct function or block[1] if user defined blocks and subroutines are not instrumented, or if only some are. This can occur for "run time" and "flushing time". All other information such as "blocked time", "send volume", *etc.*, are allocated correctly regardless of whether the functions entry and exit are instrumented because the trace records contain the information required to allocate the data.

The information is collected at the level of the sub-program, so that if subroutine "A" calls subroutine "B", the time spent in subroutine "B", the data it sent or received, *etc.*, is not reflected in the information for sub-

1. In addition to code constructs, such as do loops, that are detected by the instrumenter, AIMS provides a special set of functions that can be used to define blocks by inserting calls in the source code before instrumentation.

routine "A". The information for user defined blocks is accumulated in higher level blocks up to the subroutine containing the blocks. For example, if subroutine "A" contains a block "B" which contains blocks "C" and "D", as shown below:

```
SUBROUTINE A
    recv w bytes
    BLOCK B
        recv x bytes
        BLOCK C
            recv y bytes
        BLOCK D
            recv z bytes
```

NTV will show that "D" received z bytes, "C" received y bytes "B" received (x+y+z) bytes and "A" received (w+x+y+z) bytes

The displays that present profiling information by function show all the functions and user defined blocks in the program. If their entries and exits are not instrumented the display will show an execution time of zero since there will be no trace records indicating entry and exit.

6.0 Getting Started

The command `ntv` will began execution. There are no command line arguments. When **NTV** has loaded it will present a main window that has three buttons:

File:	brings down a pull down menu that contains buttons to exit **NTV** or to open a file selection window used to select the trace file to be viewed.
Display:	brings down a display selection pull down menu continuing buttons to select the type of displays to be presented (profiling based on functions, profiling based on processors, or time line).
Help:	produces a help display

The file selection box is a standard Motif file selection widget and operates in the standard manner. When an *AIMS* trace file is selected for loading, a window appears that will indicate that **NTV** is setting up to read data. (Sometimes this window will remain blank.) The size of the trace file is the major factor controlling how long the window stays up. If the trace file is small you may not see it. During the time this window is displayed the trace file is being examined to determine memory allocation requirements. When this has completed, the reading of the trace file begins. A window will appear giving the percentage of the file which has been read. If the trace file is small, this window may not stay open long enough to be read. For *SP2* traces the first window will not appear or will only flash because the information for memory allocation is determined during the conversion to *SDDF*.

If a program executing on an SP2 is terminated before it completes execution, the trace buffers are flushed. For these cases it is possible that there are messages that have been sent and not received or, because of clock synchronization errors, messages received that were not sent. For these cases, **NTV** will display a list of the incomplete messages in the xterm from which **NTV** was started. The list will contain, for each message, the sending node and time (if known), receiving node and time (if known), and the message tag. This can be very useful in cases where the program hung and execution had to be terminated. When the file has been loaded, all windows except for the *main* window will disappear and you can select the view of the data to be displayed.

7.0 Acknowledgments

I wish to express my appreciation to Robert T. Hood, Charles E. Niggley, and Robert L. Hirsch for reviewing the manuscript and suggesting numerous improvements. I also want to thank David C. DiNucci and Robert Hood for the many discussions we had, and numerous suggestions they made, during the development of **NTV**.

Appendix

NTV is coded in C and C++. It has been built and tested using GNU gcc and g++, but there is nothing special in the **NTV** code that limits it to these compilers. **NTV** uses the *SDDF standalone Library* from the *Pablo Performance Analysis Environment*[1]. The *SDDF standalone library* does have some restrictions which may limit the compiler choices. This could limit the compiler choices used in building **NTV**. **NTV** uses Motif Release 1.1 or higher.

The pre-processor for *SP2* traces, **ntvSP2preProc**, has two requirements that make it easiest to build on an *RS6000* based system. First, the preprocessor requires include files that describe the ECOFF file format used by *AIX*. These files are similar to the include files for the COFF format, but differ in some important ways. The difference of importance to **NTV**

1. The Pablo Performance Analysis Environment is available from the University of Illinois. It is copyrighted by The University of Illinois Board of Trustees but is freely available to non-commercial users.

occurs in the `sym.h` file. Second, `VT_trc.h`, the include file describing the *MPL* trace format (included with the **NTV** distribution), uses a definition of `timestruc_t` that may be different from that used on some UNIX systems. In particular, AIX uses:

```
        struct timestruc_t {....}
```

while at least some Unix systems use

```
        struct timestruc  {....} timestruc_t
```

Were it not for these two requirements, **ntvSP2preProc** could be built on any UNIX system.

more