PVM Monitoring with Xab

Adam Beguelin
adamb@cs.cmu.edu
School of Computer Science
Carnegie Mellon University
and
Pittsburgh Supercomputing Center

 Supercomputing 1993

Xab is a tool for Analyzing and deBugging PVM programs. It was originally developed by Adam Beguelin a ORNL and the University of Tennessee for PVM version 2. Version 3 of Xab has been developed at Carnegie Mellon University and the Pittsburgh Supercomputing Center by Adam Beguelin with the help of his students: Erik Selberg, Erik Seligman, and Wilson Swee.

Xab: X Analysis and deBugging

Goals:

- **Runtime monitoring**
- **Runtime debugging**
- **Postmortem performance analysis**

 Supercomputing 1993

Xab allows the PVM programmer to monitor PVM calls while the program is running. The results of selected PVM calls are displayed in the Xab window. The display of these calls can be useful for debugging at runtime.

Information regarding the program.'s PVM calls can be saved to a file, called a tracefile. This tracefile can be redisplayed in Xab after the program has completed, i.e. post-mortem. The tracefile can also be used by other parallel performance tools to provide different views of the runtime performance of the program.

Xab Approach

- Instrument all PVM calls (events)
- Provide runtime display of events
- Provide runtime analysis of events
- Interface to performance analysis tools
  - Paragraph
  - Pablo

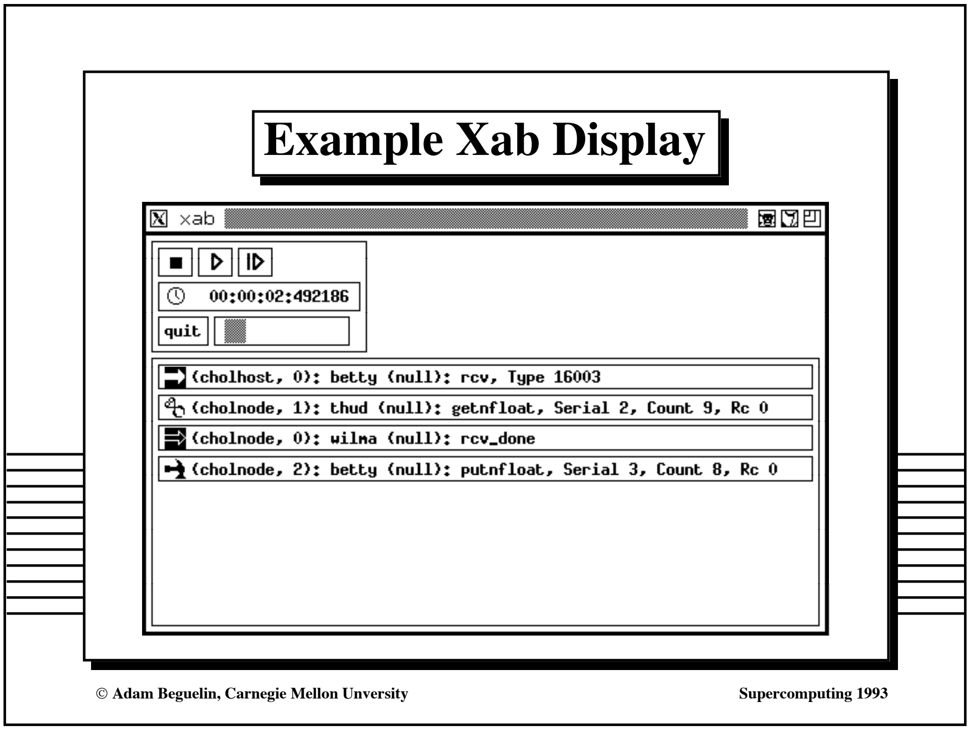
 Supercomputing 1993

In Xab all PVM calls are instrumented□. This means special monitoring code is inserted into each PVM call. This monitoring code generates events whenever PVM calls are made. An event is simply a record of interesting information about the PVM call. For instance, an event may consist of the type of PVM routine being called, the time it was called, and the result code of the call.

Xab events can be displayed as the program is running using a tool, xab, which shows the last PVM called for each task in the virtual machine.

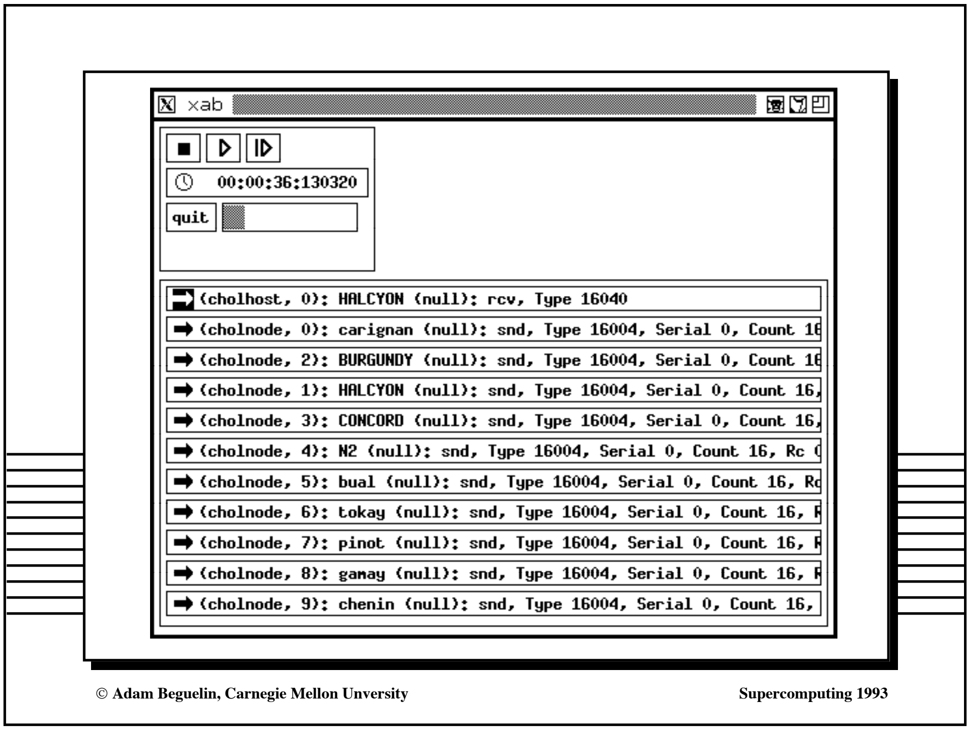
The system provides some minimal analysis of events which consists of formatting□ the event for display and adding iconic tags to the event. Other types of analysis may be added in the future. For instance, message format checking could be performed which would issue warnings when messages are unpacked using different data types than those used to pack the message.

Other packages provide a plethora of displays for viewing events generated from parallel programs. Paragraph and Pablo are such tools. Currently there is a script, topg, which will convert xab tracefiles to PICL format which is used by Paragraph. Currently we are working to make Xab tracefiles usable with Pablo as well. Use of Xab tracefiles with Paragraph is only postmortem.



This is the Xab display for version 2 of xab. Here you can see four lines, one for each PVM task. The icon on the left of each line graphically depicts the operation being performed by the PVM call. In this case the PVM task (cholhost, 0) is doing a receive, (cholnode, 1) is removing data from a recv buffer, (cholnode, 0) has just completed a receive, and (cholnode, 3) is placing data into a message buffer.

The controls on the top line control the playback of events: stop, play, and single step. The second line is a clock denoting the elapsed time for the computation. On the third line is a button for exiting the program and a slider for controlling the playback speed of the display.



This display shows a PVM program that has blocked. Each node has sent a message to the host but the host is blocked on a recieve of the wrong type.

To catch this kind of error event buffering must be disabled. If the cholhost task was buffering events it may not send the snd event that shows the error since it may never get to the point where it will flush the event buffer.

How instrumentation works

- **Each PVM call replaced by Xab call**
- **Xab sends event messages to monitor**
  - **Each call generates at least one event**
  - **Some calls generate two events**
    - » **pvm_recv()**
    - » **pvm_spawn()**
    - » **pvm_barrier()**
- **Monitor formats and displays events**

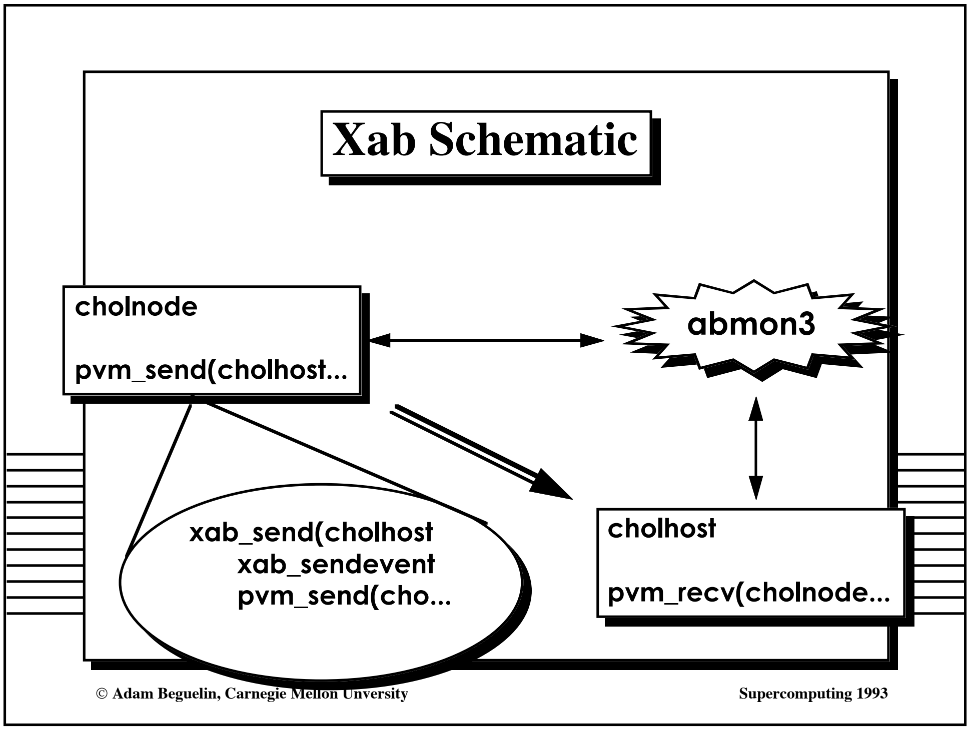
 Supercomputing 1993

Xab instruments PVM programs by replacing each PVM call with an Xab call. The corresponding Xab call will generate Xab events as well as perform the PVM call.

The Xab calls generates events by sending messages to a monitor process. These event messages are simply PVM messages containing the information pertaining to the event being generated. Thus the Xab uses the same PVM mechanism for communicating□ as does the user's program.

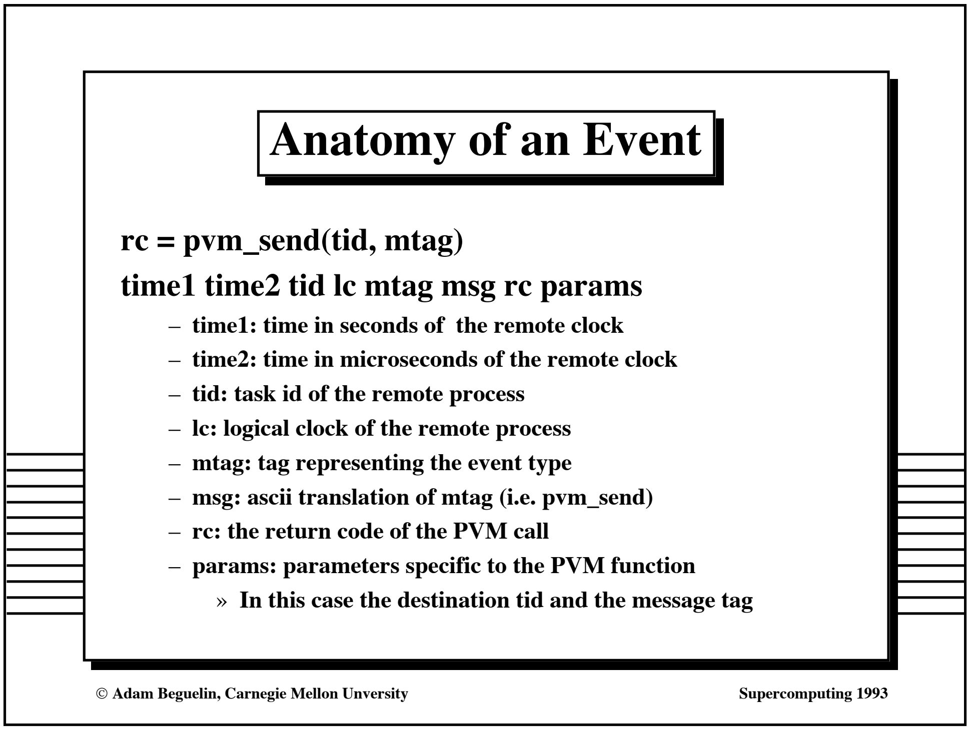
In general, each PVM call generates a single event. In some cases PVM calls may take a long time to complete and in this case it is useful to have two events, one at the beginning of the call and another at the end. For instance, the pvm_recv() call will block until a PVM messge has arrived. The amount of time spent waiting for a message is important. Thus Xab will send out an event before entering the pvm_recv() and then send out another event when the pvm_recv() has completed. The pvm_spawn() and pvm_barrier() calls are handled in a similar manner.

The monitor process is itself a PVM task called abmon3. This task receives Xab event messages, formats them, and saves them to a file or to standard output. The events can be directly piped to the Xab display tool, called xab (or xab3 for version 3).



This slide shows he cholesky demo as it is being monitored by Xab. The cholnode task is calling pvm_send to send a message to the cholhost task. Cholnode's pvm_send actually translates into an xab_send. The xab_send sends an event message to abmon3 and then performs the actual pvm_send on behalf of the program.

The double headed arrows from abmon3 to the tasks in the cholesky demo program indicate that information flows in both directions. In version 3 of Xab it is possible to alter the monitoring parameters of a PVM program as it executes□. This is done through the abmon3 program.



Event records contain information about the PVM call, where it was made, and when it was made. The order of the information an an Xab event is fairly arbitrary. The msg field is added to make it possible for a human to read the tracefiles and make some sense of them.

Instrumenting PVM Programs

C programs

- **Add include file xab3.h**
- **Link to libxab3.a**

Fortran programs

- **Link to libxpvm3.a**
- **Link to libxab3.a**

 Supercomputing 1993

The replacement of all PVM calls with Xab calls is done differently in Fortran and C. In C the programmer must add the include file xab.h or xab3.h to any source files which make calls to the PVM library. The program then must be recompiled and linked with libxab3.a.

In Fortran we have created a special library which directly calls the Xab calls rather than the PVM calls. Fortran programs simply need to link to libxpvm3.a in place of libfpvm3.a. The Fortran program must also be linked to the libxab3 library, called libxab3.a.

Running□ Instrumented Programs

- Start PVM
- Run abmon3
  - abmon3 -o <fn>
  - abmon3 | xab3
  - abmon3 | tee <fn> | xab3
  - abmon -l
- Run program

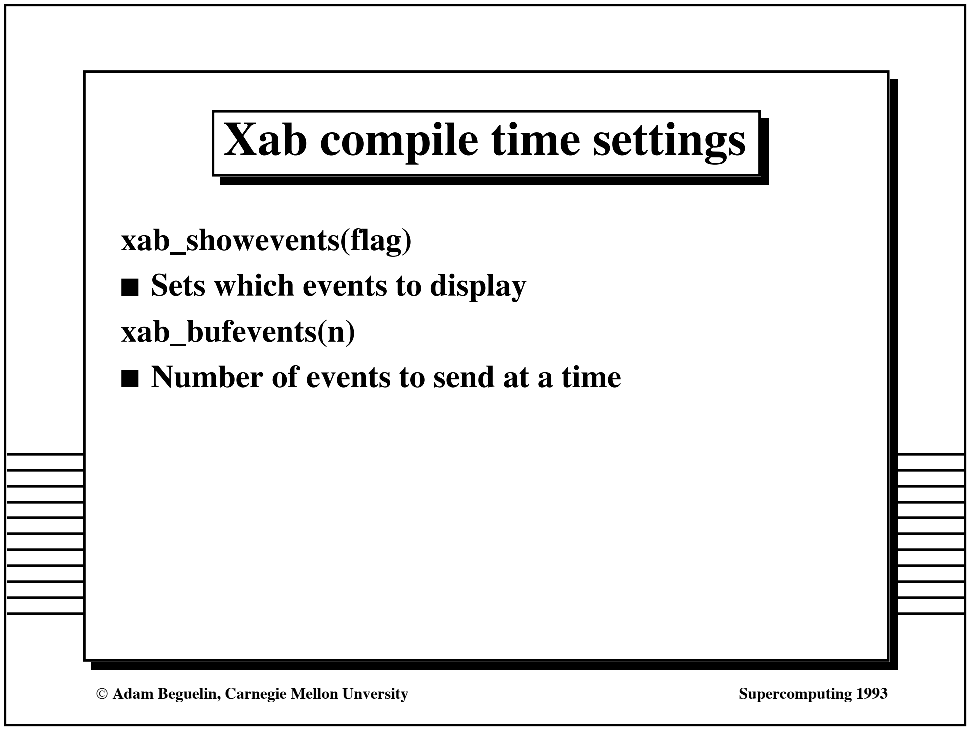
 Supercomputing 1993

Running a PVM program that has been instrumented with Xab is fairly simple. Of course PVM must be running before the program can be run.

To use Xab the monitor must also be started **before** the program is executed. On any machine in the current virtual machine, start the abmon3 process (abmon for version 2). The abmon process can be started in several modes. If abmon is given a filename it will place all the event records in that file. If no file name is given then abmon will write events to standard output. Using a UNIX□ pipe this event stream can be piped into the xab3 program which will display the events in an X window. If both a window and a file are desired then the UNIX□ tee command can be used to create the file and send the event stream to the X window display tool.

By default abmon3 will exit after the program it is monitoring has completed. If abmon3 is started with the -l flag then abmon will live even when all the tasks which it is monitoring have exited. This is useful if you want to monitor multiple runs of a program without having to restart the monitor.

Once the monitor is running, the instrumented PVM program can be executed.



It is possible that not all PVM calls need to be monitored. It is possible to set a flag which will tell Xab to only send certain events to the monitor. Since Xab events travel over the same network as program data it is advised to only monitor only necessary events. The function xab_showevents() will tell Xab to only show certain kinds of events. See the Xab documentation for a full explanation of the different values for the flag parameter to xab_showevents(). The call to xab_showevents() must be manually placed in the user program if values other than the default settings are desired. It is also possible□ to alter this setting at runtime using xabsf from anywhere in the virtual machine.

Although sending an event as soon as it happens can result in timely monitoring, it can also congest the network. To avoid this congestion it is possible to tell xab to buffer events before sending them. The xab_bufevents call will accomplish this. If the number of events is set to 10 then a monitor message (containing 10 events) will only be sent every 10th event. If the program halts because of an error, buffered events will not be sent to the monitor. If the program exits normally and there events which have yet to be sent to the monitor, they will be flushed at this time.

Xab runtime settings

- **xabsf - xab set flags**
- **Sets the events currently monitored**
- **Sets the buffering of events**

xabsf -pC -E 4 -t 40019 4001a 180004

- Turns on pack and common
- Sets buffering to 4 events per message
- Effects only the tids listed

xabsf -h

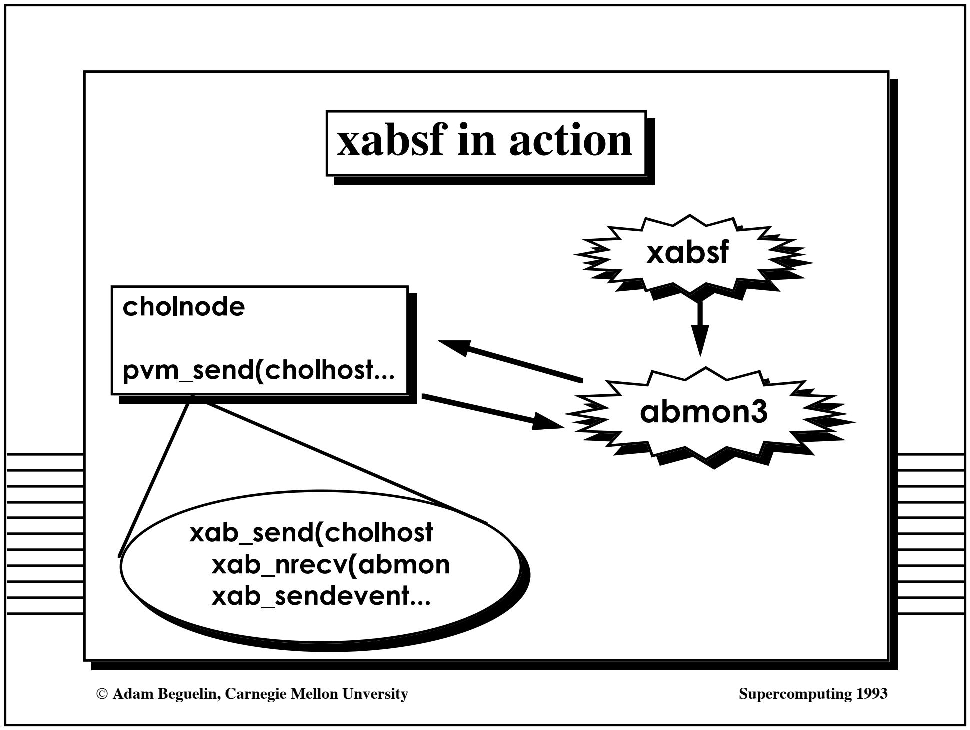
- Describes the options to xabsf

 Supercomputing 1993

It is possible to alter the monitoring parameters of a program as it is running. This can be done through the xabsf command. The command can be typed from any machine in the virtual machine. It can alter the type of events currently being monitored (ala xab_showevents) and the number of events being sent in each monitor message (xab_bufevents).

This command works by sending a control message to abmon3. The abmon3 task then sends control messages to the user tasks currently being monitored. Note that the effect is not instantaneous. There may be many event messages buffered in the system and xabsf only throttles back events by changing flags in the user's tasks.

xabsf routes the control messages through abmon3 since it is the only task known and trusted by the user's tasks.



The xabsf command adjusts the monitoring parameters of a running program by first sending a message to abmon. The abmon task then sends a control message to the task which whose level of monitoring needs to be adjusted.

Every Xab call checks for control messages before actually doing the corresponding PVM call. It is significant that control messages only come from abmon3. If control messages were sent directly to the user's program by xabsf it would be impossible to distinguish control messages from user messages. It is important that Xab does not consume user messages! This is accomplished by assuring that Xab only consumes messages which come from abmon3, thus are guaranteed to be control messages and not user messages.

Tachyons: Impossibly ordered events

- Xab does no clock synchronization
- Logical clocks may be incorporated
- xab postprocessor will remove tachyons

 Supercomputing 1993

Xab places timestamps on events. These timestamps are taken from the local clock on each processor. Xab does not attempt to synchronize clocks. We are currently considering implementing a logical clock scheme in Xab. Such a scheme has been implemented in a prototype of Xab but it was found not to be portable. We have designed a new more portable scheme but have yet to implement it in the current version of Xab.

If you are using Xab for PVM version 2.x you may use a post processor, called xab-post, which will remove tachyons from Xab tracefiles. The xab-post program is also available on netlib in pvm/xab.

Generating Paragraph Tracefiles

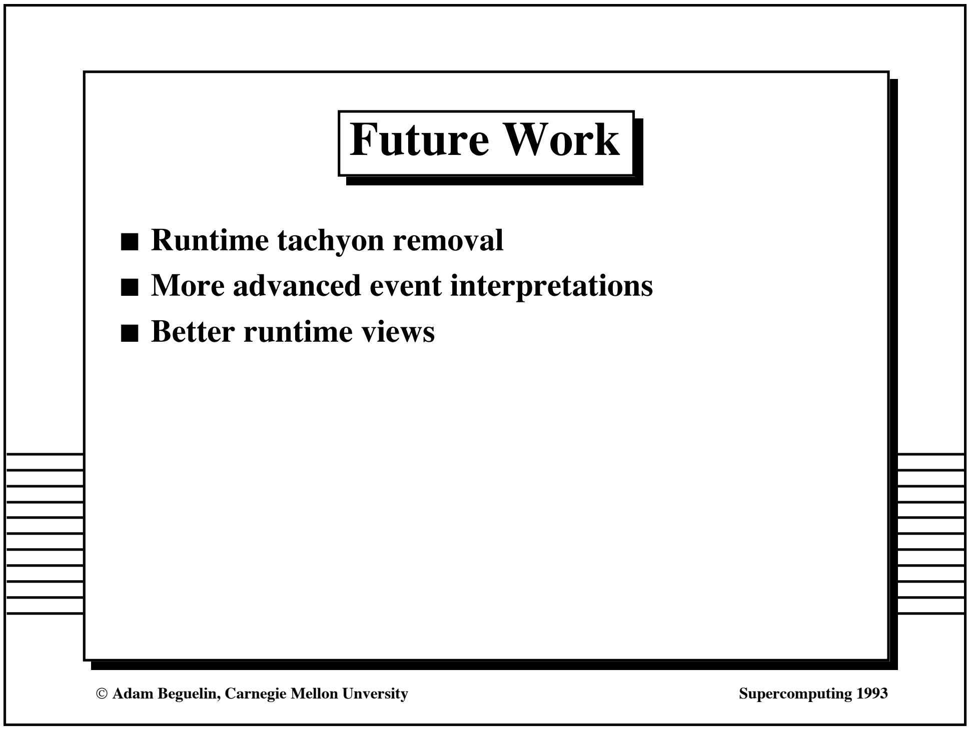
- **Create Xab tracefile**
  mytrace.xab
- **Run topg script**
  topg mytrace
  - **Assumes mytrace.xab**
  - **creates mytrace.trf**
- **Tachyons will not be removed**
- **PICL node numbers used**
  Not PVM tid or component name

 Supercomputing 1993

Assuming you have created the tracefile with the extension .xab you can convert it to a PICL tracefile by running the script topg and giving the name of the tracefile (without the extension) as a parameter. topg is a shell script that uses an awk script to convert the tracefile to PICL format. PICL tracefiles typically end in .trf so topg creates a file ending in .trf.

Paragraph expects events to be properly ordered. Xab and topg do not ensure events are properly ordered. They are simply ordered by the timestamps taken from the local computer where they were generated. If your machines do not have synchronized clocks then you can either use xab-post or run the program on a single machine where clock synchronization is not a problem.

Paragraph labels processes as *node* numbers where PVM names them as task ids or by component name and pid. The topg script will arbitrarily translate PVM names to Paragraph node numbers.



It would be possible to add facilities to the monitor, abmon3, which would adjust the timestamps of events to insure no tachyons exist. The cost involved in this scheme would be two fold: delays would be introduced into the runtime monitoring since certain events would not be displayed until their corresponding preceding□ event had been processed; the storage and CPU utilization for the monitor process would be increased.

Events could be processed and warnings issued for situations which are likely to be the result of program errors. For instance messages which have been sent but not received□ should be flagged. Another example would be messages which are nondeterministic, i.e. race conditions within the program.

Finally, better runtime views would be beneficial. Many user’s have expressed the desire to be able to step backwards as well as forwards through a tracefile, even at runtime.

Xab availability

- **Send mail to netlib@ornl.gov**
  - **Version 2: Subject "send index from pvm/xab"**
  - **Version 3: Subject "send index from pvm3/xab3"**
- **Bug reports or questions**
  - **comp.parallel.pvm**
  - **xab@psc.edu**
  - **pvm@msr.epm.ornl.gov**

 Supercomputing 1993

Versions of Xab exist for versions 2 and 3 of PVM. Version 3 of Xab is actively being developed but it's functionality will eventually be incorporated into PVM and XPVM.

Your comments and questions are welcome.