Performance Visualization Tools

Performance Visualization Tools

Lecture Outline : Following Topics will be discussed

- Characteristics of Performance Visualization technique
- Commercial and Public Domain tool on PARAM 10000
- Example programs to demonstrate Selective tools
  - Public Domain Tool : UPSHOT
  - Sun Cluster Tool :PRISM
  - C-DAC HPCC Software

Performance Visualization Tools

A programmer should begin the investigation into causes of performance degradation with high level and abstract views, so that global trends can be seen. Suggested guidelines are:

- Dissatisfied with performance?
- Compare progress with ideal behavior?
- Examine deviation in overall behavior
- Examine individual sections/PE
- Low level Investigation

3

Performance Visualization tools

- Delivery of adequate performance is not automatic and performance evaluation tools are required in order to help the programmer to understand the behavior of a parallel programs
- Do I have a performance problem at all?
  - Time/HW counter measurement
  - Speed up and Scalability measurements
- What is the main bottleneck (Calculations, communication, synchronization)?
- Where is the main bottleneck?
  - Call Graph profiling (`gprof`)
- Indicate what parts of your program are consuming more resources. Gives detailed idea about optimization efforts and analyzing the performance of programs.

4

Profiling and Tracing

- Profiling
  - Summary information is recorded (misses; calls; time)
  - About program entities (Basic blocks, functions)
  - Bottlenecks in various fragments of the code
  - Good in-sight into performance data
- To enable profiling, you must recompile and relink using "`-p`" flag.
  - `cc test.c -p –o -c`
  - `cc junk.c -p –o -c`
  - `cc test.o junk.o-p-o run`
  - Creates '`run`' binary which is ready for profiling
- `gprof :` It replicates the abilities of `prof`, plus it gives a call graph profile so you can see who calls who, and how often.
- Tracing
  - Record information about events

5

Profiling vs Tracing

(Contd…)

- Profiling
  - Recording summary information (time, #calls, #misses..)
  - About program entitles (functions, objects, basic blocks)
  - Very good for quick, low cost overview
  - Points out potential bottlenecks
  - Implemented through sampling or instrumentation
  - Moderate amount of performance data
- Tracing
  - Recording information about events
  - Trace record typically consists of timestamp, processid..
  - Output is a trace file with trace records sorted by time
  - Can be used to reconstruct the dynamic behavior
  - Creates huge amounts of data
  - Needs selective instrumentation

6

Common Performance Problems with MPI

- Frequent synchronization
  - Reduction operations
  - Barrier operations
- Load balancing
  - Wrong data decomposition
  - Dynamically changing load
- Handle Dead-lock free programs on message passing clusters
- Debugging MPI programs

7

Common Performance Problems with MPI

(Contd…)

- **Synchronization Speed**

  Refers to the time needed for all processors to agree they have finished one step of a problem and are ready to go on together to the next step.

- **Synchronization methods**
  - Waiting all processes finish a loop
  - Waiting until the first of any of the contributing processes finds a particular answer
  - Assigning a unique task to each processor from a list of tasks

8

Characteristics of Performance Visualization Tools

Evaluation of tools public or commercial tools are based on the following parameters.

- Accuracy ; Simplicity ; Flexibility
- Intrusiveness
- Abstraction
- Robustness
- Usability
- Scalability
- Portability
- Versatility

9

Characteristics of Performance Visualization Tools

(Contd…)

- **Intrusiveness :** We need to be aware of following overheads and account for it when analyzing data
  - Synchronization among message passing clusters introduces quite lot of overhead
  - Performance analysis data collection inevitably introduces some overhead.
  - Overhead in hardware and software component of the cluster must be aware of.
- **Abstractness**
  - A good performance tool allows data to be examined at a level of abstraction appropriate for the programming model of parallel program.

10

Characteristics of Performance Visualization Tools

❖ **Robustness** (Contd…)

- Errors should be handled by displaying appropriate diagnostic messages and should be indicated in a file
- Tool should not cause the user to stuck when he/She takes a wrong action by mistake and give proper feedback
- The installation procedure should be as simple as possible
- Handle debugging and error handling for improper behavior of tool

❖ **Usability**

- To be useful, a tool should have adequate documentation and support, and should have an intuitive easy-to use interface
- Adequate functionality should be provided to accomplish the intended task without putting undue burden on the user to carry out low-level tasks, such as manual insertion of calls

11

Characteristics of Performance Visualization Tools

❖ **Usability** (Contd…)

- The tool should support SPMD/ MPMD programming paradigms.
- A performance view must be useful in the process of understanding parallel programs behavior
- Incorporation and integration of state of the art technologies including application domain expertise, human-computer interaction (HCF) visual perception and graphic design
- Easy way to represent the causes of performance degradation with high-level and abstract views, proceeding from the highest level to the lowest level.
- While a collection of views, animations and general performance data can help uncover performance bugs, some guidelines or strategies and needed to direct the order in which views are examined

12

Characteristics of Performance Visualization Tools

❖ **Scalability** (Contd…)

- For scalability, we look for the ability to handle large numbers of processes and large or long-running programs.
- The problem size and the machine size is increased to study the performance anomalies, keeping Amdahl's law and Gustafson's law for scalability.
- Scalability is important both for data collection and for data analysis and display.
- For data collection, desirable features are scalable trace file formats, a mechanism for turning tracing on and off, and filtering of which constructs should be instrumented.
- For data analysis and display, important scalability features include ability to zoom in and out, aggregation of performance information, and filtering.

13

Characteristics of Performance Visualization Tools

❖ **Portability** (Contd…)

- Most parallel programmers will work on a number of platforms simultaneously or over time.
- Portability is a relative measure, which depends on both the language used, compiler used and the target machine with various system area networks, compilers and the system software
- Programmers are understandably reluctant to learn a new performance tool every time they move to a new platform.
- Thus, we consider portability to be an important feature. For portability, we look for whether the tool works on most major platforms and for what MPI implementations and languages it can handle.
- The tool must be easily usable when compilers, system area networks, and the message passing libraries are changed and should produce correct behavior.

14

Characteristics of Performance Visualization Tools

(Contd…)

- **Versatility**
  - For versatility, we look for the ability to analyze performance data in different ways and to display performance information using different views.
  - Providing a unified view of the profiling data generated by each process buy modifying the time stamps of events on different processes so that all the processes start and end at the same time and providing a convenient form for visualizing the profiling data in a *Gantt chart, Bar Chart, Histograms and Graphs.*
  - Another feature of versatility is the ability to interoperate with other trace formats and tools.
  - Use appropriate method to visualize and profile the parallel program data to give quick feed-back to user regarding performance issues

15

MPICH Tracing

- MPICH distribution provides portable MPI tracing library through multiprocessing Environment.
- MPI programs generate traces when compiled with
  - MPI log option to mpicc, mpiCC, mpi f 77 or mpi f 90
- http://www.fp.mcs.ant.gov/~lusk/upshot/

16

Performance Visualization Tools

- Tracing of MPI Programs
  - Upshot/Jumpshort (MPICH)
  - MPICL/Paragraph
  - AIMS, NTV
- Profiling / Tracing of parallel, multi-threads programs
  - TAU's Portable Profiling and Tracing Package
- Dynamic Object Code instrumentation
  - Dyninst
  - DPCL
- Automatic Performance Analysis
  - Paradyn
- Portland Group of Compilers
- Totalview

17

Performance Visualization Tools

(Contd…)

- Vampir (Pallas)- Commercial product of Pallas, Germany
  - Visualization and Analysis of MPI programs
  - Offline trace analysis for message passing trace files
  - Detail information about point-to-point and Collective communication messages with color display
  - Profiling (Execution time, no. of calls, I/O operations, communication statistics
- Vampir trace (Pallas)
  - Library of Tracing of MPI and application events
  - Record user subroutines
- Dimemas Pallas
  - Analyze load-balance and communication dependencies
  - Performance issues related to bandwidth, message contention
- Guide view (KAI) : Performance Analysis of Open MP programs

18

Performance Visualization Tools : UPSHOT

- Upshot
  - Upshot is an X-based visualization tool for performing post-mortem analysis of logfiles produced by the alog package.
  - It provides one type of view of a logfile, in which events are aligned on the parallel time lines of individual processes
  - It also provides a convenient form for visualizing the profiling data in a *Gantt chart*.
  - For example : Visualization of an MPI program that computes PI value using MPI point-to-point communication calls using Upshot is as follows:-

19

Performance Visualization Tools : UPSHOT

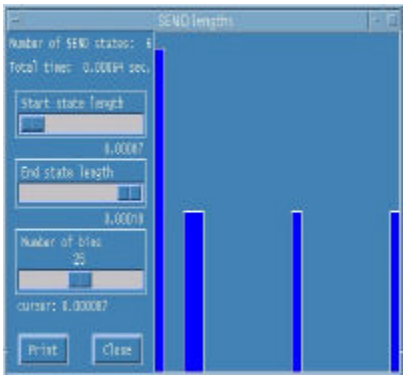
- Gantt Chart View :- It shows events that are aligned on the parallel timelines of individual processes, with processes on Y-axis and time on X-axis. Each states are represented by one color.

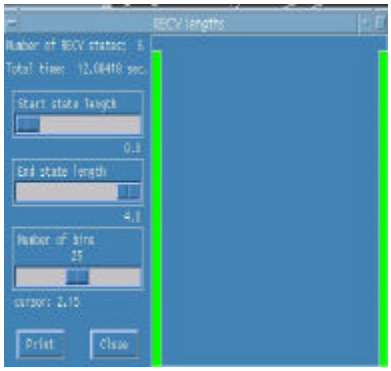
20

Performance Visualization Tools : UPSHOT

- Histogram representation of individual states which give total time taken by all processes for each particular state and start and end state lengths.

MPI_Send **MPI_Recv**





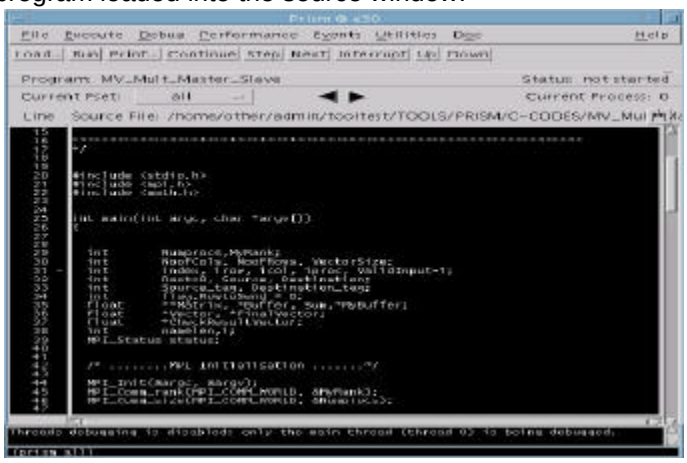
Performance Visualization Tools : PRISM

- Prism is a Sun HPC tool
  - The Prism programming environment is an integrated graphical environment within which users can develop, execute and debug programs.
  - It provides an easy-to-use, flexible and comprehensive set of tools for performing all aspects of serial and MPI programs.
  - It helps to determine how efficiently the various parts of your Sun MPI program run and where program's performance can be improved
  - For example : Visualization of MPI program that computes Matrix-Matrix Multiplication using Master-Slave(Self Scheduling algorithm using Prism.

Performance Visualization Tools : PRISM

(Contd…)

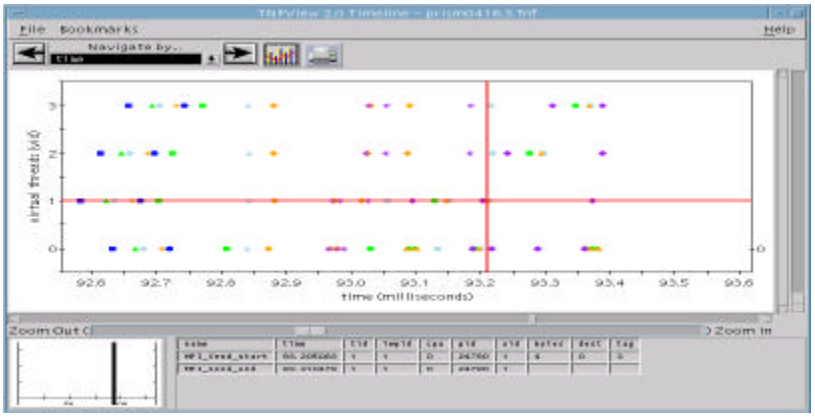
- The Prism Programming Environment's Main Window with program loaded into the source window.



Performance Visualization Tools : PRISM

(Contd…)

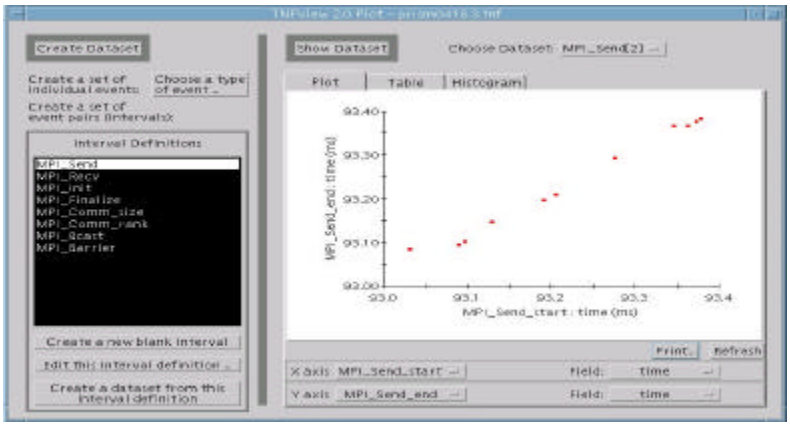
- Time line window which shows timeline view of the TNF probe trace data with processes on the Y-axis and time(ms) on X-axis and name of the state in Event table which is at the bottom.



Performance Visualization Tools : PRISM

(Contd…)

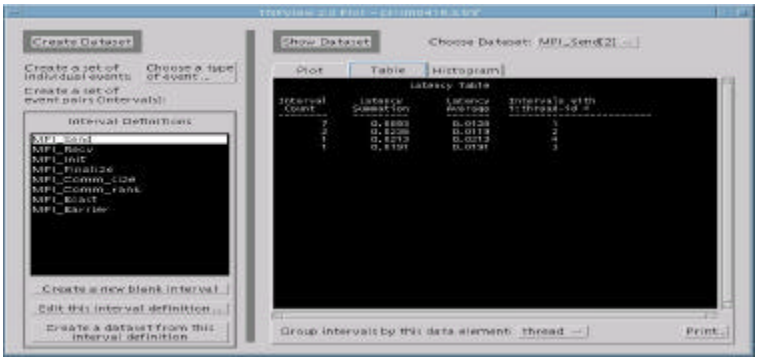
- Scatter Plot View of data set Mpi_Send with set of all Mpi_Send_start events on X-axis and set of all Mpi_Send_end events on Y-axis.



Performance Visualization Tools : PRISM

(Contd…)

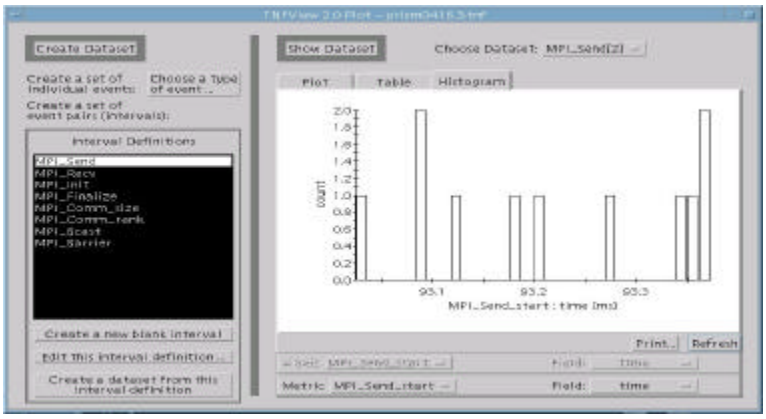
- Table View of a data set Mpi_Send which gives Interval Count(Number of intervals), Latency summation(Time in ms),Latency Average(Time in ms) and Intervals with data_element.



Performance Visualization Tools : PRISM

(Contd…)

- Histogram View of a data set Mpi_Send which shows events of data set by bars with time on X-axis and count of events on Y-axis



27

C-DAC HPCC SOFTWARE FOR UNIX CLUSTERS

- CDAC HPCC software for Unix clusters
  - High performance flexible software environment
  - Adheres to established and emerging standards in parallel computing
  - Complete solution for creating and executing parallel programs
- CDAC HPCC Software provides a complete solution on Unix clusters through
  - Base Software
  - Program Development Environment
  - System Management Tools
  - Software Engineering Tools

28

C-DAC HPCC Software : FORTRAN 90

- Conformance to language standards Validated through International Test Suites
- Robustness ensured
  - By exhaustive testing
  - Compiled over 1 million lines of real applications
- Support
  - Prompt and efficient maintenance
  - Easy software up gradations
- FORTRAN-77 to Fortran-90 Converter
- An easy and efficient way of converting the existing FORTRAN77 code to Fortran90.
  - Understanding Fortran90 in a better way.

29

C-DAC HPCC Software : DIViA

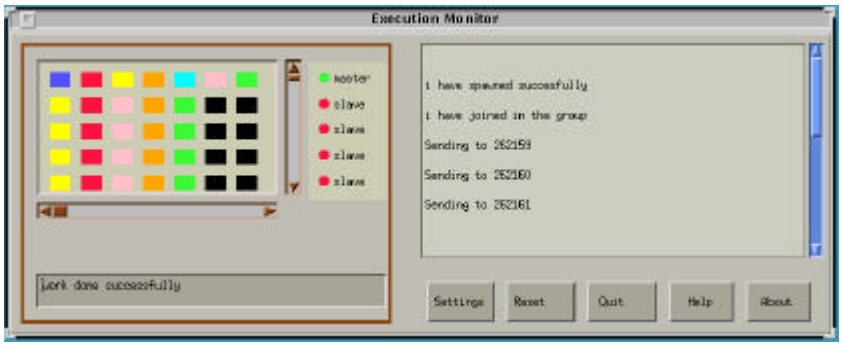
- DIViA (Debugger with Integrated Visualizer and Analyzer) is an advanced portable and flexible parallel debugging environment.
- It consists of a coherent set of tools that help programmer in both correctness and performance debugging.
- Correctness Debuggers:-
  - Multiprocess Debugger (MPD)
  - Message Debugger
  - Visual Debugger (ParViD)
  - Execution Monitor (ExMon)
- Performance Debuggers:-
  - Automatic Communication Bottleneck Detector (ABND)
  - Profile Visualizer (ProfViz)

30

C-DAC HPCC Software : ExMon

❖ **Execution Monitor (ExMon) features:-**

- Execution flow monitor of sequential as well as parallel applications.
- Visual and textual monitoring



31

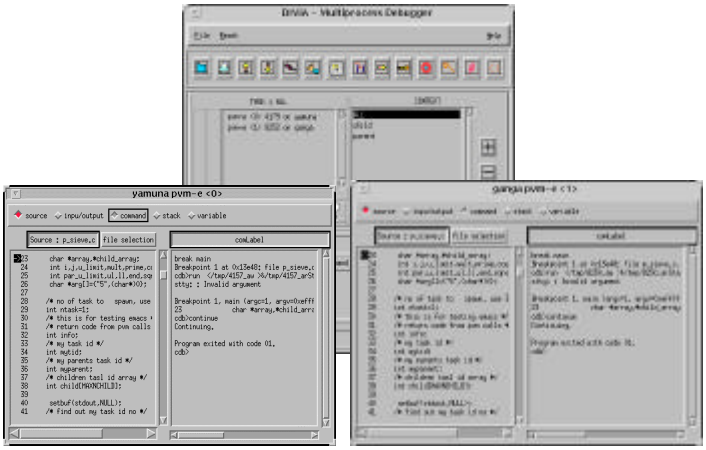
C-DAC HPCC Software : MPD

❖ **Multiprocess Debugger (MPD) features**

- To Debug Parallel Applications in any Parallel environment
- Debugging multiple processes from a single point
- Selective debugger control of dynamically created tasks
- Interactive source code debugger for individual task
- Task and group operations Debugging remote and distributed programs
- Easy to use interface
- Online help with Popups and Status bars

32

C-DAC HPCC Software : MPD



33

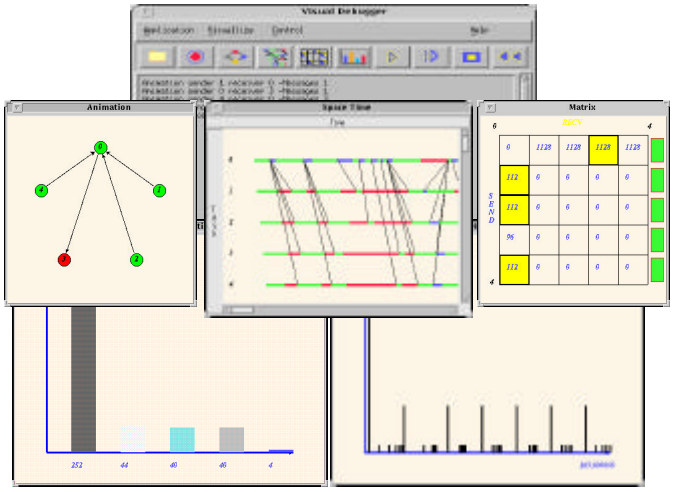
C-DAC HPCC Software : ParViD

❖ **Visual Debugger (ParViD) features:-**

- Visual Portray of various communication events in an abstract manner
- Three different visual display windows; Communication Animation, Space Time Diagram and Communication Matrix
- Focus on order of interprocess communication through Communication Animation.
- Communication status display of the constituent tasks.
- Display speed control with communication events history
- Provision for pause, single step and rewind of application.
- On-line/off-line modes of operation.
- Textual monitoring of message specifications.

34

C-DAC HPCC Software : ParViD



35

C-DAC HPCC Software : ABND

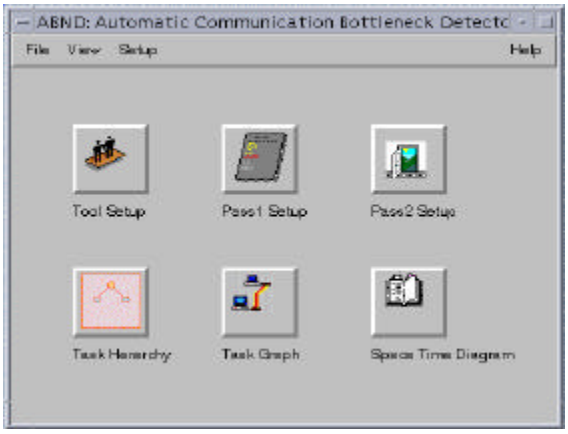
❖**Automatic Communication Bottleneck Detector(ABND) features:-**

- Automatic Communication Bottleneck Detection
- Provision to hierarchically pinpoint the actual region of bottleneck
- Task Graph to know the details of each task's communication
- Space Time Diagram to see the communication events in the bottleneck region.
- Source Code View corresponding to the event in the bottleneck region.
- Static instrumentation method to collect event traces.
- Independent trace collection and analysis

36

C-DAC HPCC Software : ABND

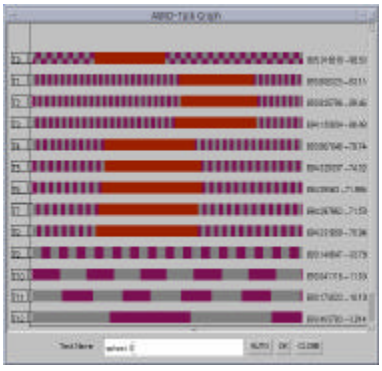
❖ ABND main GUI

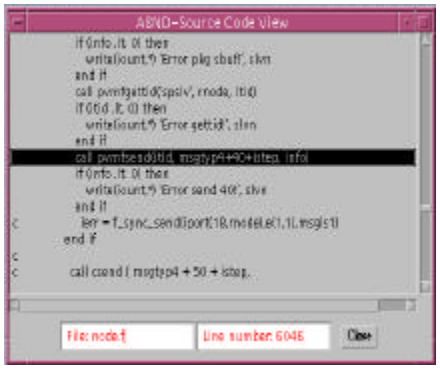


37

C-DAC HPCC Software : ABND

❖ Task Graph to know communication cost details





38

C-DAC HPCC Software : ProfViz

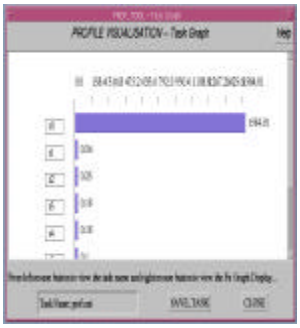
❖**Profiler Visualizer (ProfViz) features:-**

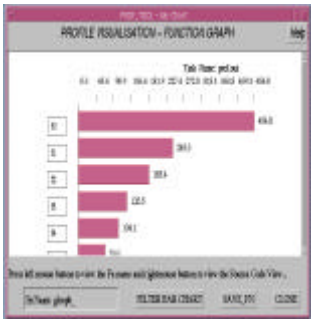
- Hierarchical data representation.
- Different performance analysis views using Task Graph,
- Gantt Chart and Source Code View
- User selectable filters to mask irrelevant data

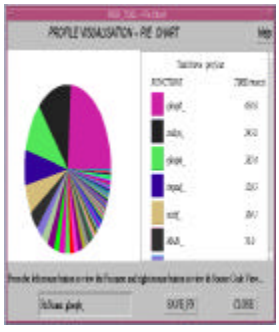
39

C-DAC HPCC Software : Profile Visualizer

Task Graph : Function Graph Pie Chart







40

C-DAC HPCC Software : System Management Tools

❖**System Management Tools : PARMON**

- Provides SSI at application level (System Management)
- Familiar to Unix users; easy-to-remember names (e.g., p<unix-command-name>)
- Scalable: fast enough to use with the same regularity that users use *ls* and *ps*, regardless of the number of nodes
- Can monitor Process Activities; Kernel Activities; CPU Parameters; Memory Parameters;Disk Parameters; Network Parameters; Physical and logical views

41

Performance Visualization Tools

Conclusions

- Performance Visualization tools for parallel programs have been discussed.
- Usefulness of public domain tools such as UPSHOT and commercial tools such as PRISM and C-DAC HPCC Software with examples have been discussed

42

Thank you

43