Tuning and Analysis Utilities for Portable Object-Parallel Programming

Darryl Brown
Steven Hackstadt
Allen Malony
Bernd Mohr

Department of Computer and Information Science
University of Oregon
Eugene, Oregon 97403
{darrylb,hacks,malony,mohr}@cs.uoregon.edu

Supported in part
by the National Science Foundation under grant NSF ASC92-13500,
by ARPA under Rome Labs contract AF 30602-92-C-0135
and Fort Huachuca contract ARMY DABT63-94-C-0029,
by a grant from OACIS, and by an IBM R&D contract MHVU3704.

Motivation

- High-level parallel language compilers (HPF, pC++, CC++, ...) generate code
  - node programs in conventional Fortran or C++
  - with runtime system library calls using message-passing (MPI, PVM, p4, RSR, ...) or shared memory operations

- **Problem**: program and performance analysis
  - ➠ currently used analysis tools (ParaGraph, XPVM, ...) show mainly low-level message-passing
  - ➠ don't show relation to source code
  - ➠ doesn't help the **user** in program optimization

- Necessary **requirements** for high-level parallel language programming environments
  - Providing a user (program-level) view
  - Support for high-level, parallel programming languages
  - Integration with compilers and runtime systems
  - Portability, extensibility, and retargetability
  - Usability

➠ **Our solution**: τ (TAU: ***T***uning and ***A***nalysis ***U***tilities)

pC++ – The Project

- ARPA funded projects

  "*Languages, Libraries and Performance Evaluation Tools for Scalable Parallel Systems*"

  and

  "Programming Environments, Compiler Technology and Runtime Systems for Object-Oriented Parallel Processing"

- Project partners
  - D. Gannon, Indiana University
    - pC++ compiler and runtime system, Sage++ toolkit
  - A. Malony, J. Cuny, University of Oregon
    - TAU program analysis tools, Ariadne debugger
  - D. Grunwald, W. Waite
    - ELI parser generator, compiler optimizations

<u>pC++ – Supported Systems</u>

- ❑ Shared memory systems
  - ❍ Kendall Square KSR-1 / KSR-2
  - ❍ Sequent Symmetry
  - ❍ SGI Challenge + Onyx
  - ❍ BBN TC2000

- ❑ Distributed memory systems
  - ❍ TMC CM-5
  - ❍ Intel Paragon
  - ❍ IBM SP-1 / SP-2
  - ❍ Cray T3D
  - ❍ Meiko-CS2
  - ❍ Workstation Clusters with PVM (homogeneous)
  - ❍ UNIX Workstations (SUN, HP, DEC, IBM, SGI, ...)

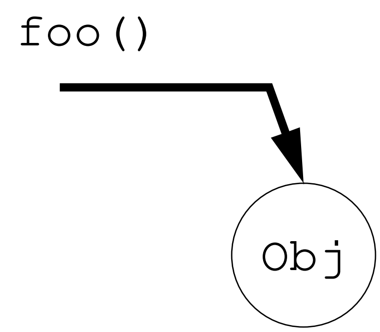
- ❑ Planned
  - ❍ Convex SPP

➠ The <u>same</u> pC++ program will run <u>without modification</u> on all platforms

pC++ – The Programming Language Ideas

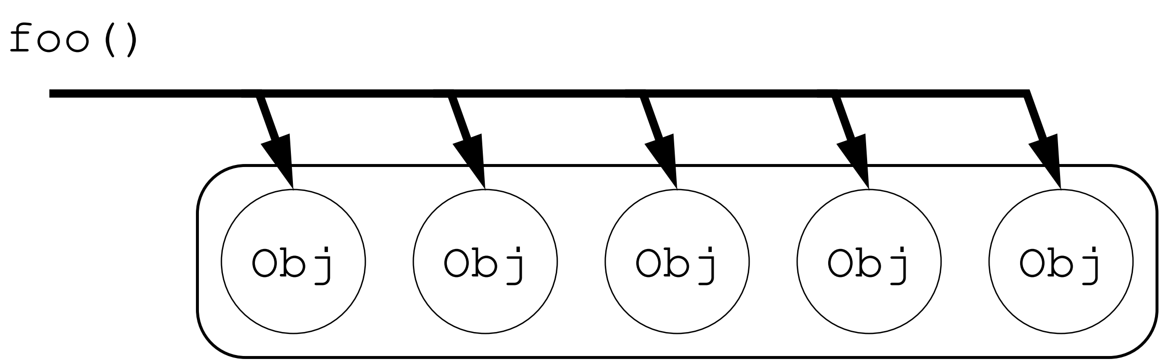
- **regular C++**: programmers apply operators and functions to objects as “messages”

```
class Obj {
    int x;
    void foo();
};

Obj myObj;

myObj.foo();
```



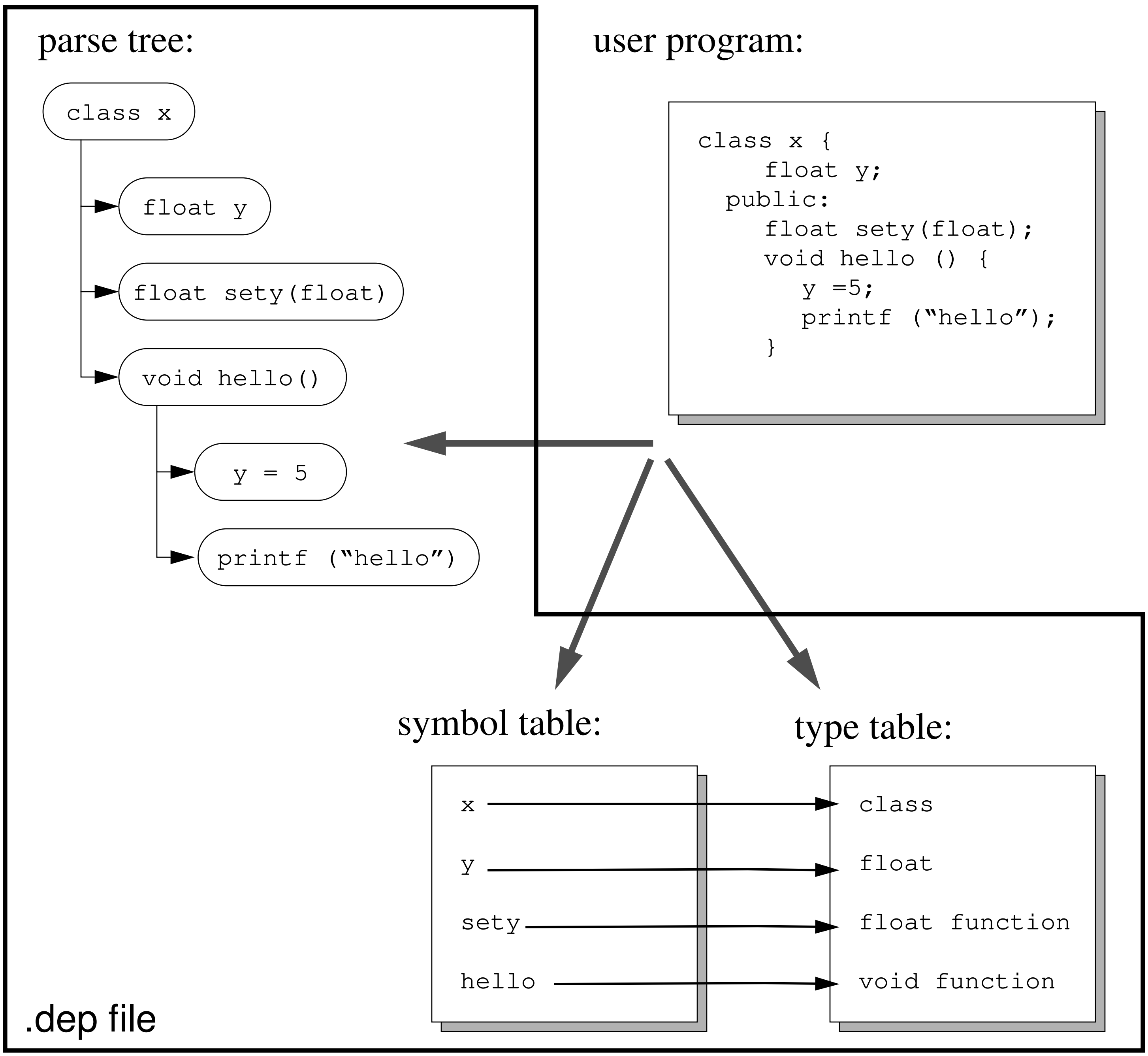
- **pC++**: this concept is extended so that an operator or function can be applied to a large set, grid, array (:= **collection**) of objects (:= **elements**) in parallel

```
Collection Vector { ... };

Vector<Obj> paraObj(AlignObj, DistrObj);

paraObj.foo();
```



Sage++

- C++ class library for building program analysis and transformation tools
- contains parsers for Fortran 77 / 90, ANSI C, and C++
- organized as a class hierarchy for accessing and modifying the parse tree, symbol table, and type table



τ

☆ Tools Are Us ☆

- ❑ Providing a user (program-level) view
  - ➠ τ graphical interface objects represent pC++ language level objects: collections, classes, methods, functions
- ❑ Support for high-level, parallel programming languages
  - ➠ τ designed and implemented in concert with pC++ system
  - ➠ τ translates low-level performance data ⇔ language level
- ❑ Integration with compilers and runtime systems
  - ➠ τ integrated with pC++ runtime system
  - ➠ τ uses Sage++ for access to ***P***rogram ***D***ata***B***ase (PDB)
- ❑ Portability, extensibility, and retargetability
  - ➠ τ implemented using C/C++ and Tcl/Tk for portability
  - ➠ τ implemented as **hypertools** for extensibility
  - ➠ τ uses Sage++ for retargetability
- ❑ Usability
  - ➠ τ tool objects represent **hyperlinks**
  - ➠ τ supports **global features**
  - ➠ τ has on-line hypertext help

τ

☆ Tools Are Us ☆

- Currently available τ tools:
  - **cosy** (***CO***mpile manager ***S***tatus displa***Y***)
  - **fancy** (***F***ile ***AN***d ***C***lass displa***Y***)
  - **cagey** (***CA***ll ***G***raph ***E***xtended displa***Y***)
  - **classy** (***CLASS*** hierarch***Y*** browser)
  - **racy** (***R***outine and data ***AC***cess profile displa***Y***)

- Prototypes:
  - **easy** (***E***vent ***A***nd ***S***tate displa***Y***)
  - **breezy** (***BR***eakpoint ***E***xecutive ***E***nvironment for visuali***Z***ation and data displa***Y***)
  - **dandy** (***D***istributed ***A***rray ***N***avigator ***D***ispla***Y***)
  - **crafty** (***C***ont***R***ol flow ***A***nd ***F***unc***T***ion displa***Y***)
  - data distribution and performance visualization (POPEYE)

- τ can work with a local or remote pC++ language system
- τ currently restricted to C++/pC++ programs
- FORTRAN (Fortran 90, HPF) support planned

τ

☆ Tools Are Us ☆

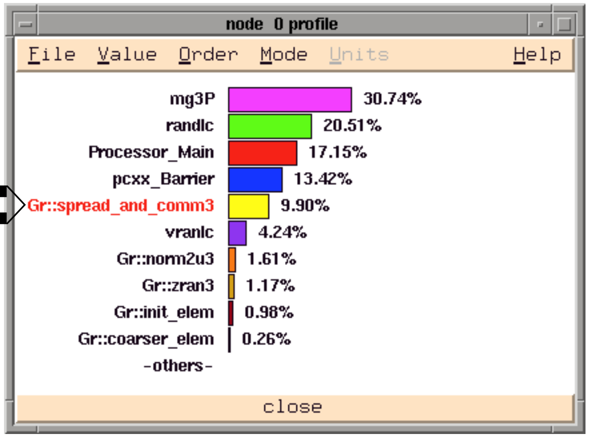
- ❑ **Hypertools**
  - ➠ τ tools are distinct tools, but they act in concert as if they were a single, monolithic application
  - ➠ implemented using **hyperlinks** and **global features**

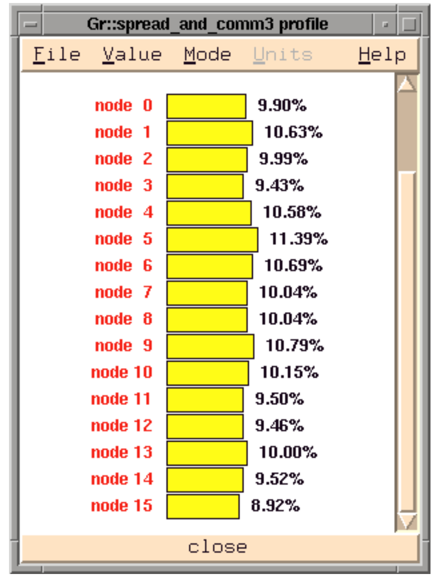
- ❑ **Hyperlinks**
  - ➠ τ graphical interface objects act like in hypermedia documents
  - ➠ selecting an object of interest brings up a more detailed or related view of the object

    e.g., selecting a class in the class hierarchy graph displays a table of class members

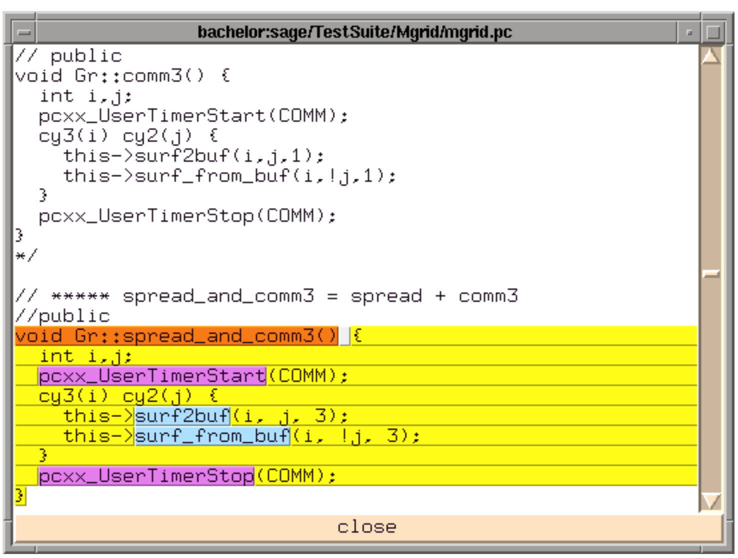
- ❑ **global features**
  - ➠ **synchronized hyperlinks**: execution of a global feature in one tool automatically updates views in the other tools
  - ➠ currently implemented:
    - ➢ `select-function`
    - ❍ `load-depfile`
    - ➢ `select-class`
    - ❍ `select-callsite`

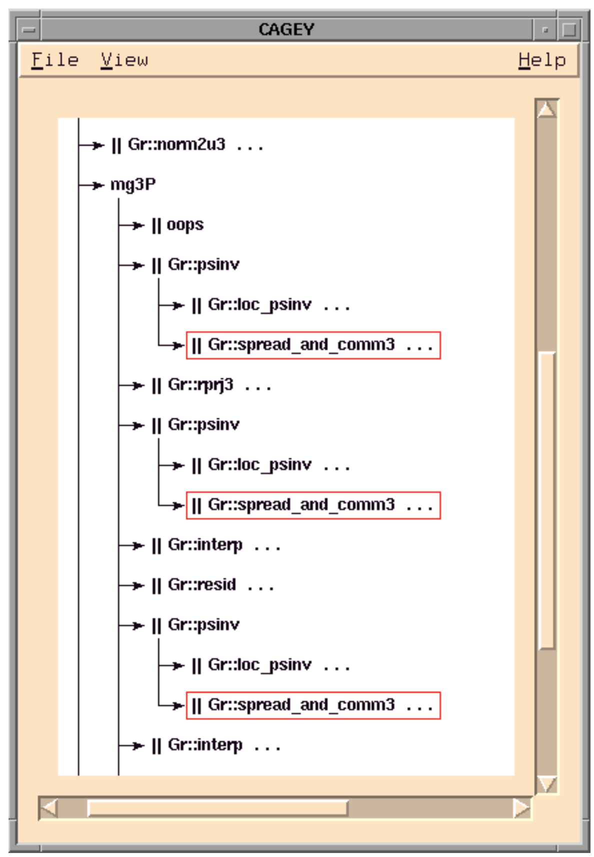
Implementation of global features





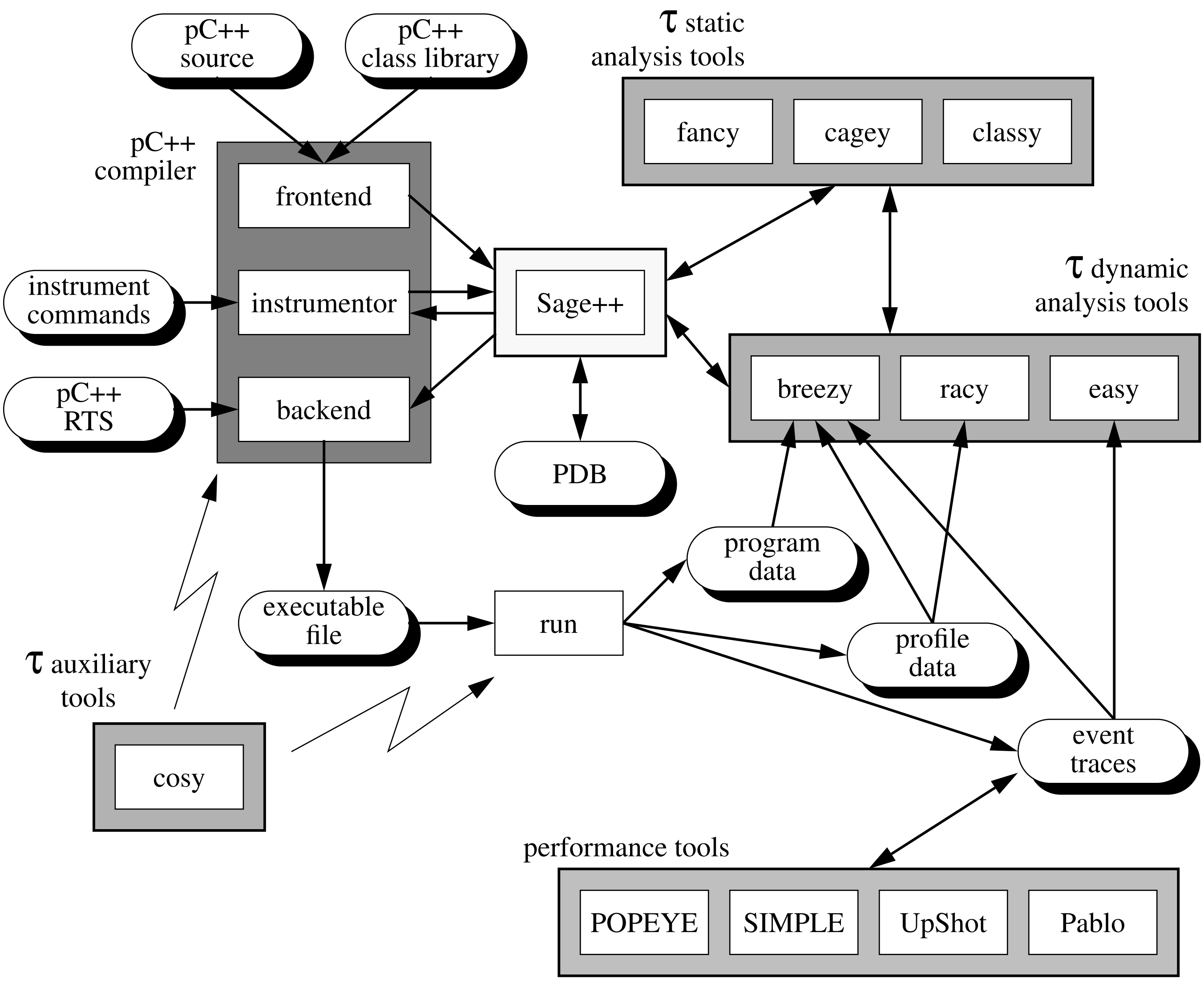

```
proc globalFeatureFoo {obj} {
   localFeatureFoo $obj
   foreach tool $FooToolList {
      if [ isRunning $tool ] {
         send $tool "localFeatureFoo $obj"
      }
   }
}
```





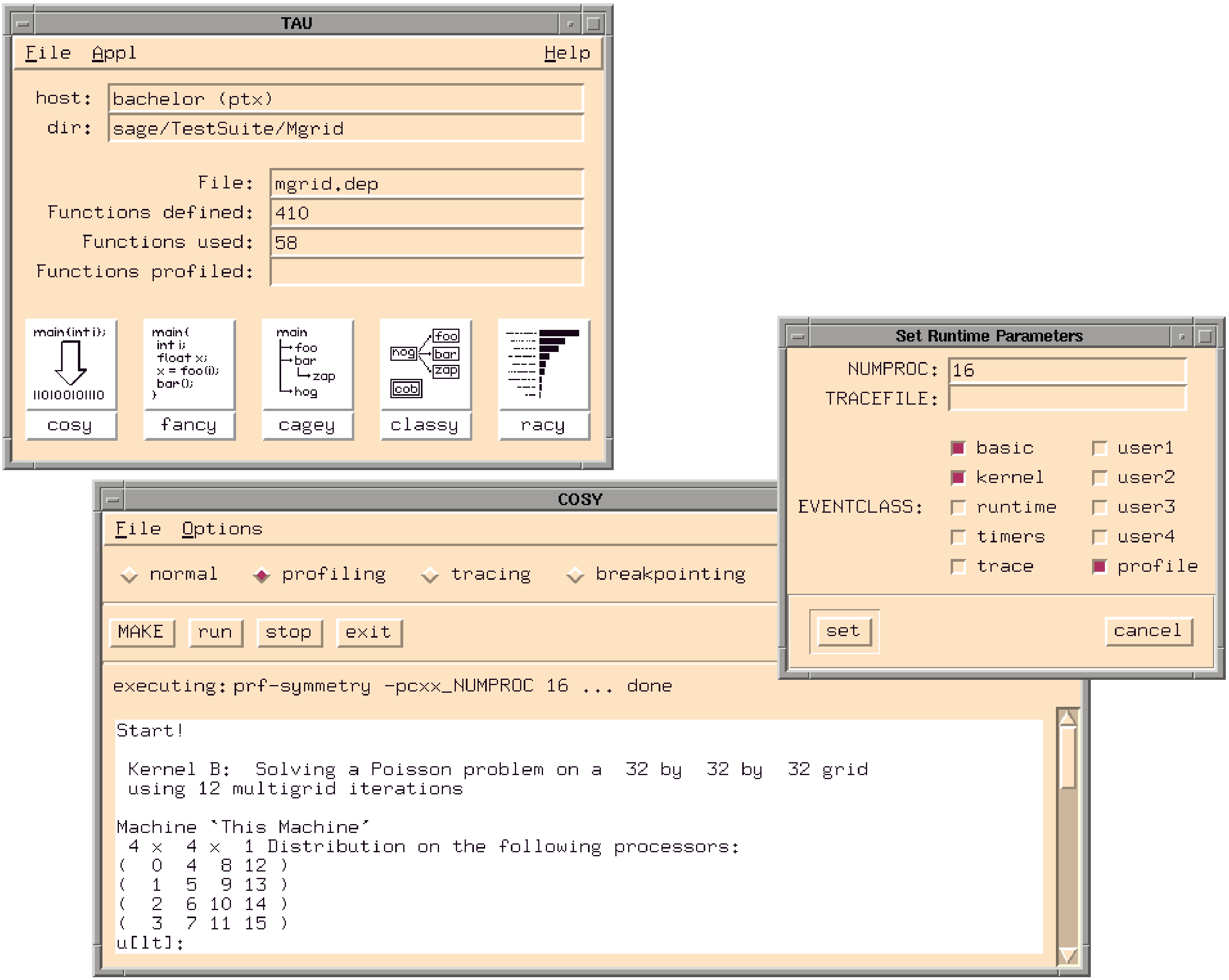
The pC++ Programming Environment

τ

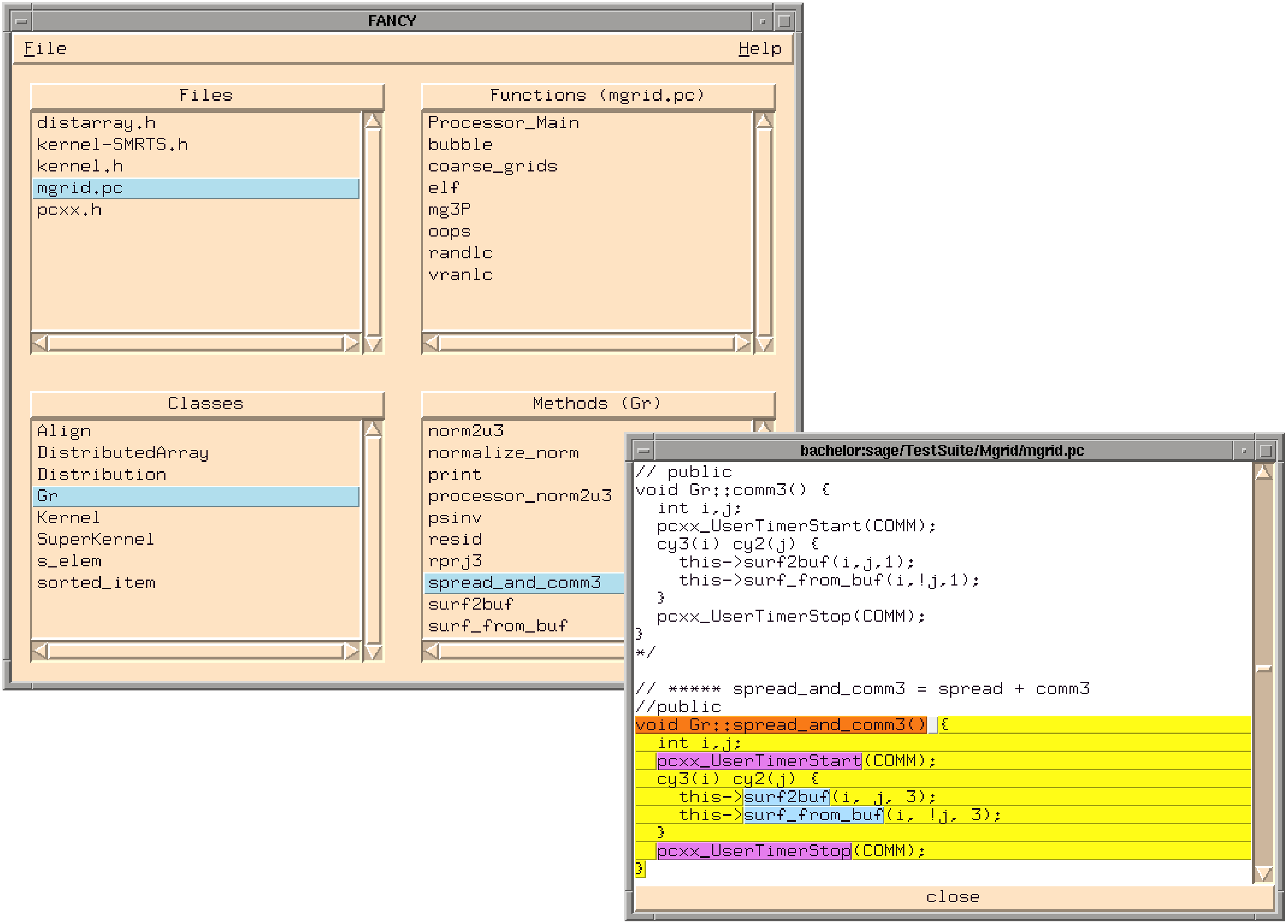


TAU / COSY
Main Control Panel / Compile and Execute Manager

τ

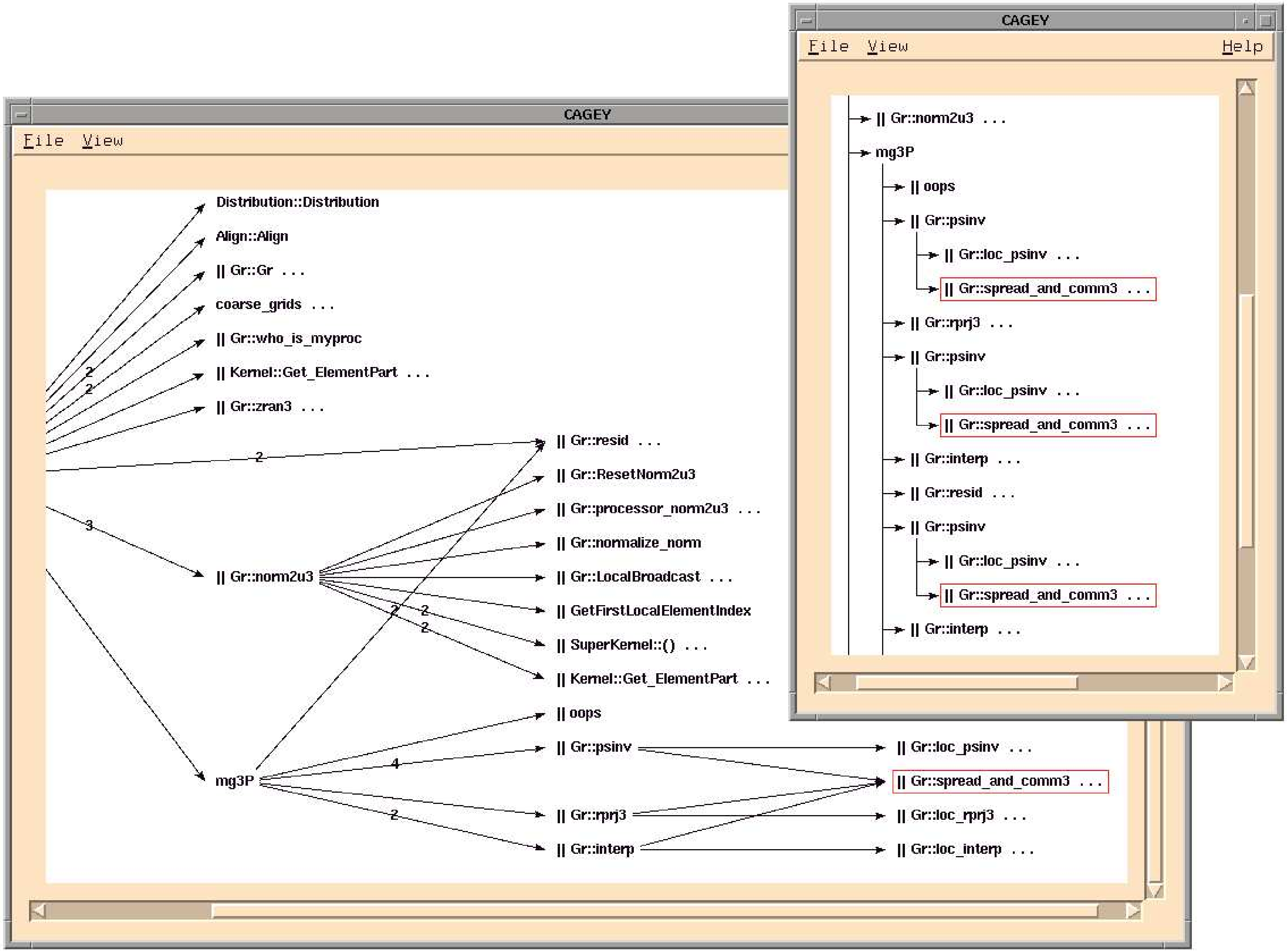


FANCY
Source Code Browser

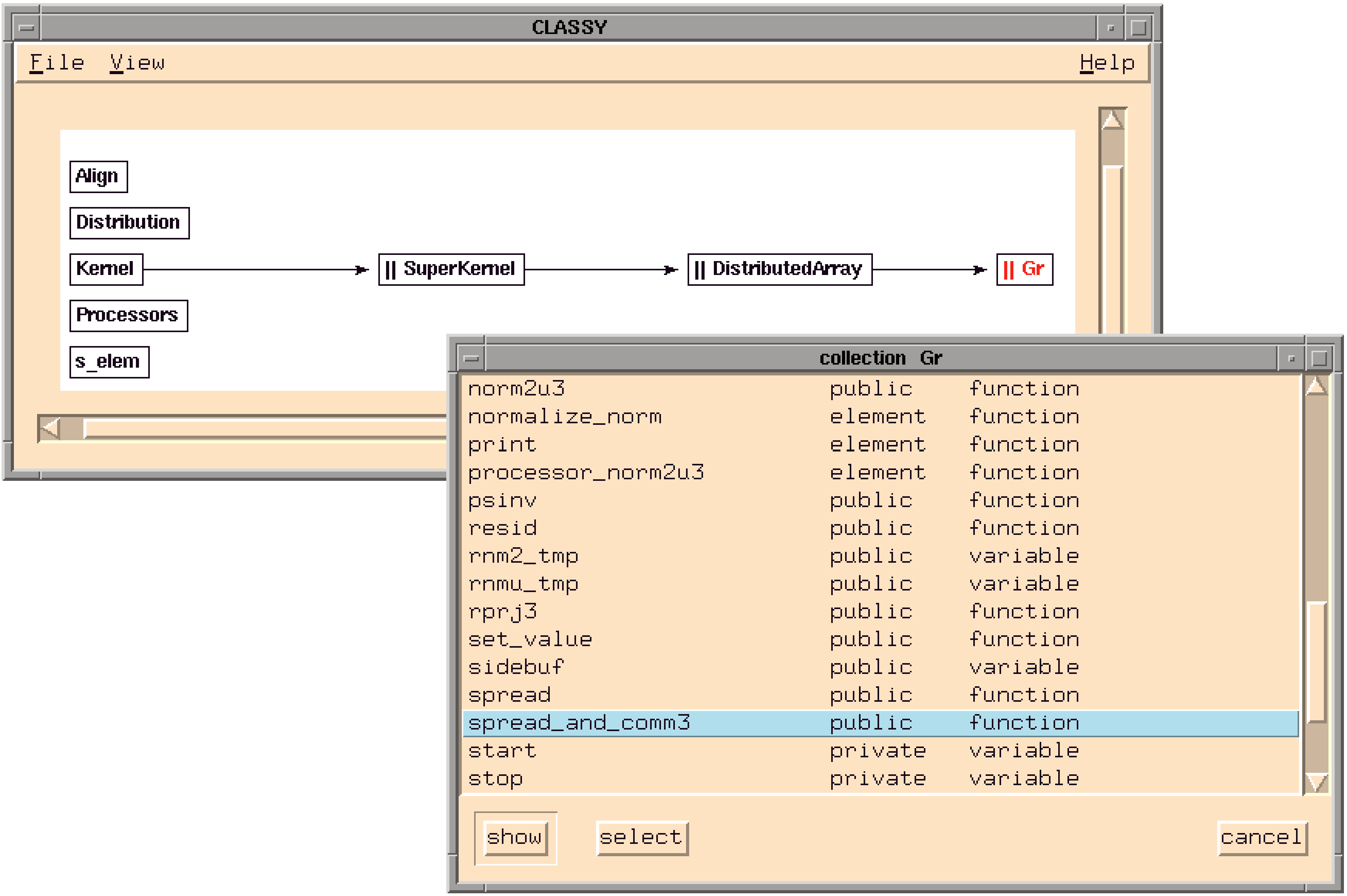


CAGEY
Static Callgraph Browser

τ



CLASSY
Class Hierarchy Browser



PPROF / RACY Parallel Program Profiler

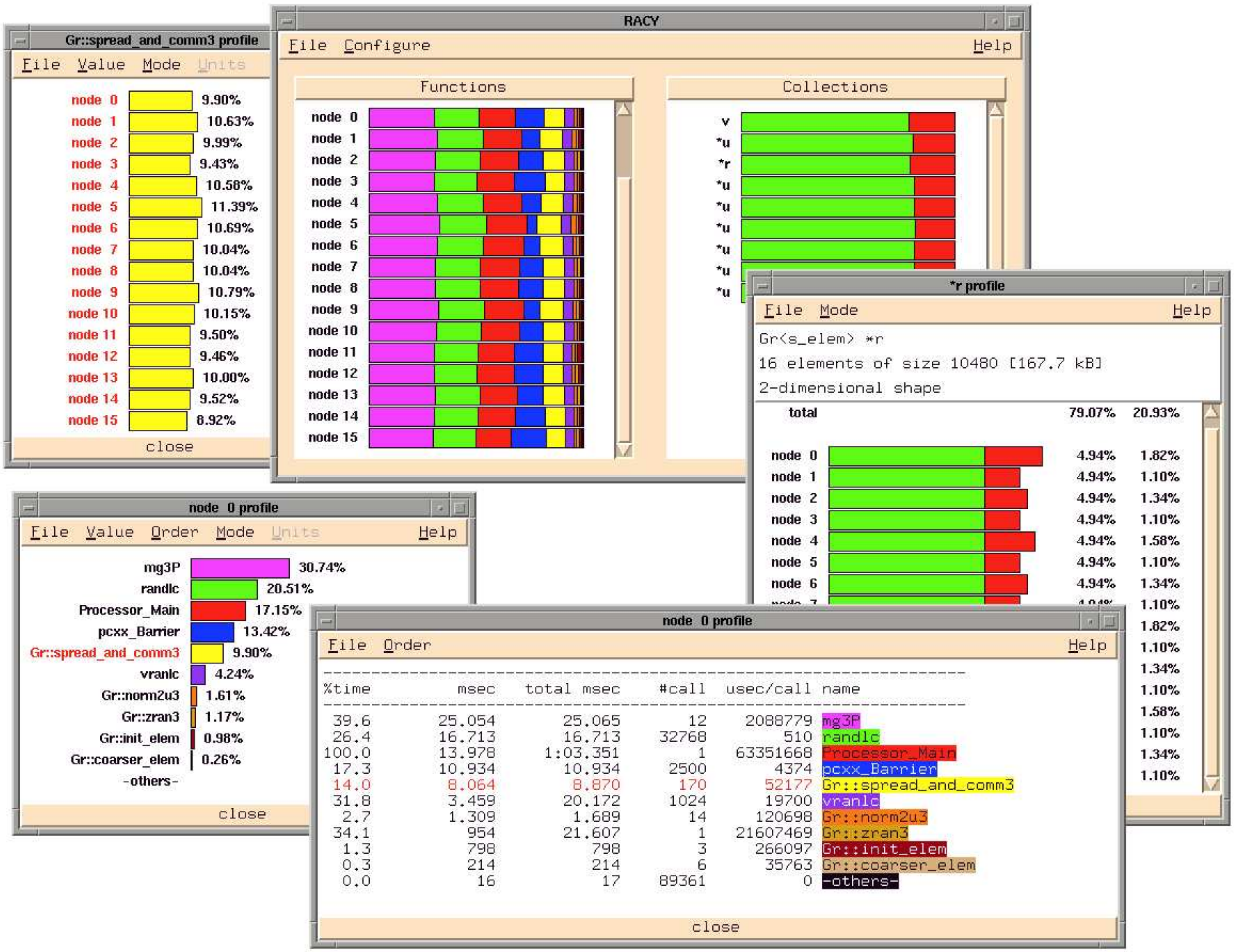
τ

- uses **instr** tool to dynamically instrument
  - entry and exit of
  - functions, class member functions, constructors, and destructors
  - in the user application and the pC++ kernel
- uses “profile-instrumented” runtime system
- data gathered for each profiled function:
  - time spent in function including and excluding its children
  - number of calls
- data gathered for each pC++ collection:
  - number of local and remote accesses per node
- profile data is stored in one file per node in a **machine-independent** format
- **pprof**: prints ASCII profile report (like UNIX’s prof)
- **racy**: graphical profile data browser

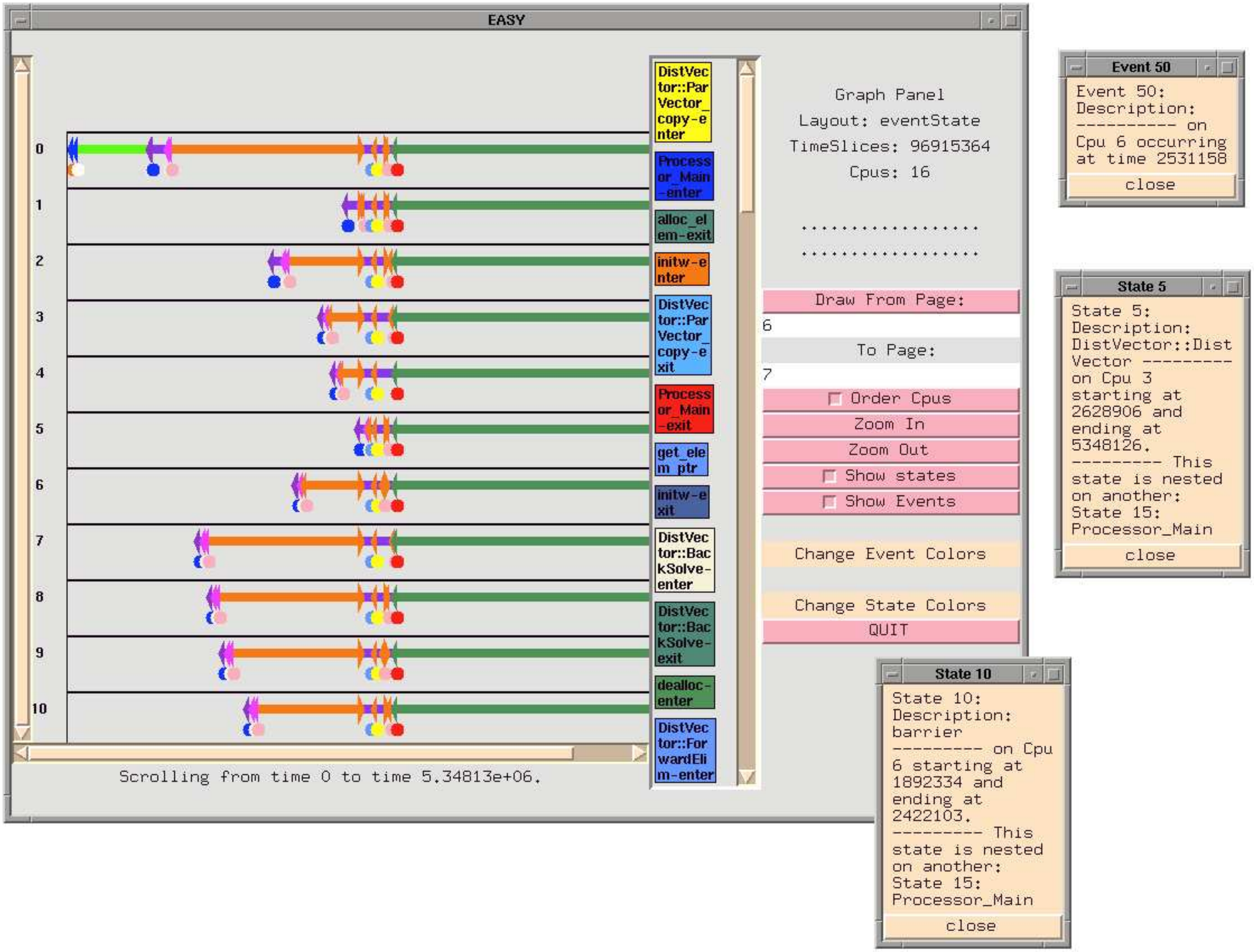
❑ Example pprof Output

```
NODE 0:
---------------------------------------------------------------
%time   msec total msec  #call usec/call name
---------------------------------------------------------------
100.0 15.761     28.869      1  28869157 Processor_Main
 14.5  4.166      4.172      1   4172459 initw
 12.2  3.512      3.512 560567         6 Vector::get
  6.9      9     2.004      2   1002201 DistVector::DistVector
  6.6  1.919      1.919     44     43628 Barrier
  5.9  1.694      1.694     32     52958 Vector::allocData
  5.5    132      1.578      1   1578414 poisson_solve
  2.9      1        835     32     26117 ParSinTransform
  2.9    638        833     32     26055 Vector::SinTransform

   local   remote collection
accesses accesses num   name
   16619      496   0   F
    9789        5   1   U

// similar output from other 31 processors deleted

FUNCTION SUMMARY (mean):
---------------------------------------------------------------
%time   msec total msec  #call usec/call name
---------------------------------------------------------------
100.0 15.312     28.301      1  28301704 Processor_Main
 14.4  4.058      4.064      1   4064819 initw
 12.4  3.519      3.519 563042         6 Vector::get
  7.9  2.246      2.246     44     51065 Barrier
  7.1      9     2.004      2   1002249 DistVector::DistVector
  5.6    116      1.578      1   1578486 poisson_solve
  4.8  1.348      1.348     32     42138 Vector::allocData
  3.0      1        845     32     26418 ParSinTransform
  3.0    640        843     32     26356 Vector::SinTransform

COLLECTION SUMMARY:
---------------------------------------------------------------
DistVector<Vector> F, collection #0
      513 elements of size 4216, 1-dimensional
      532996 local / 15841 remote accesses
DistVector<Vector> U, collection #1
      513 elements of size 4216, 1-dimensional
      282720 local /   423 remote accesses
```

RACY
Parallel Program Profile Visualizer

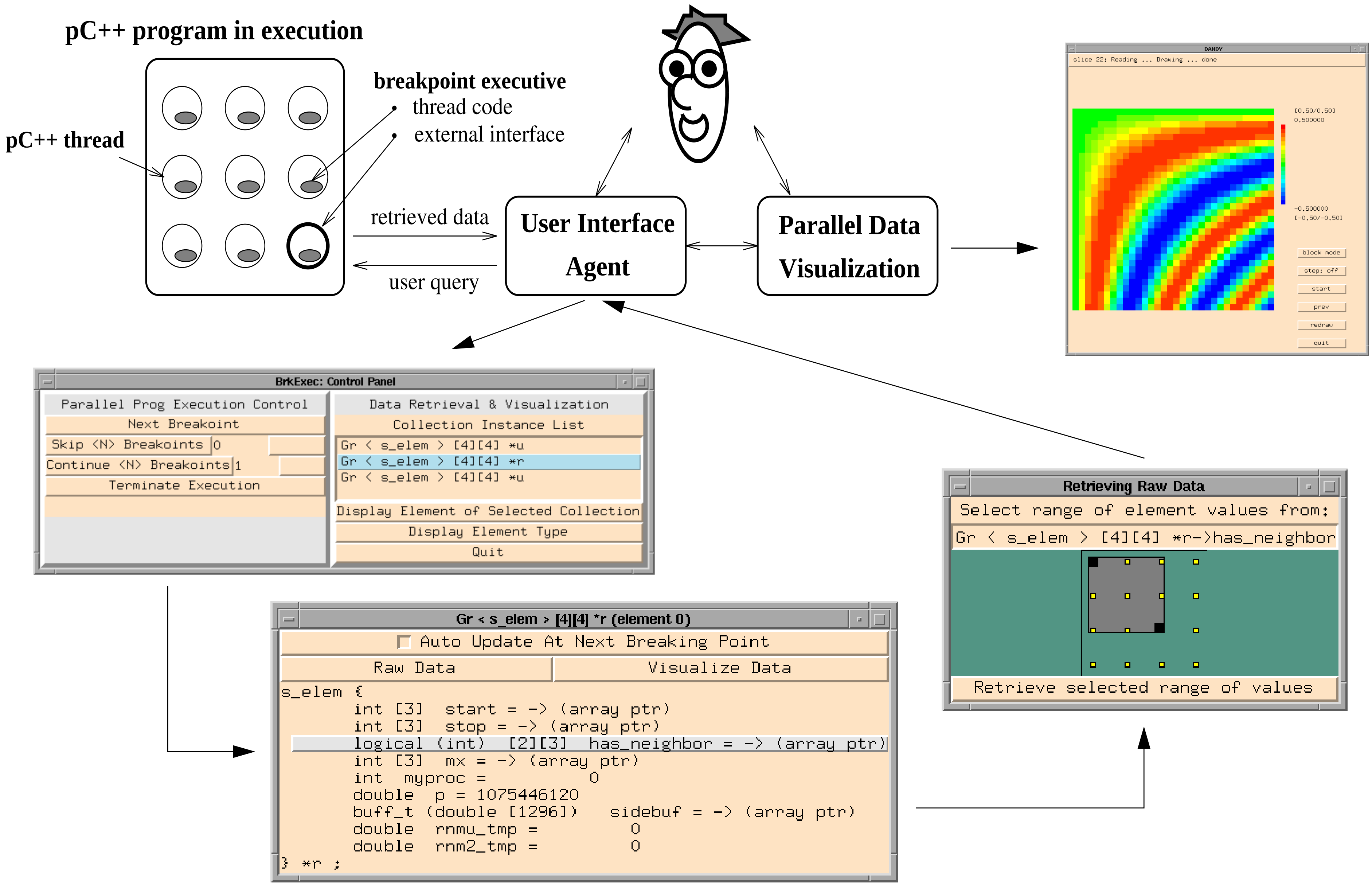


EASY
Event Trace and State Browser



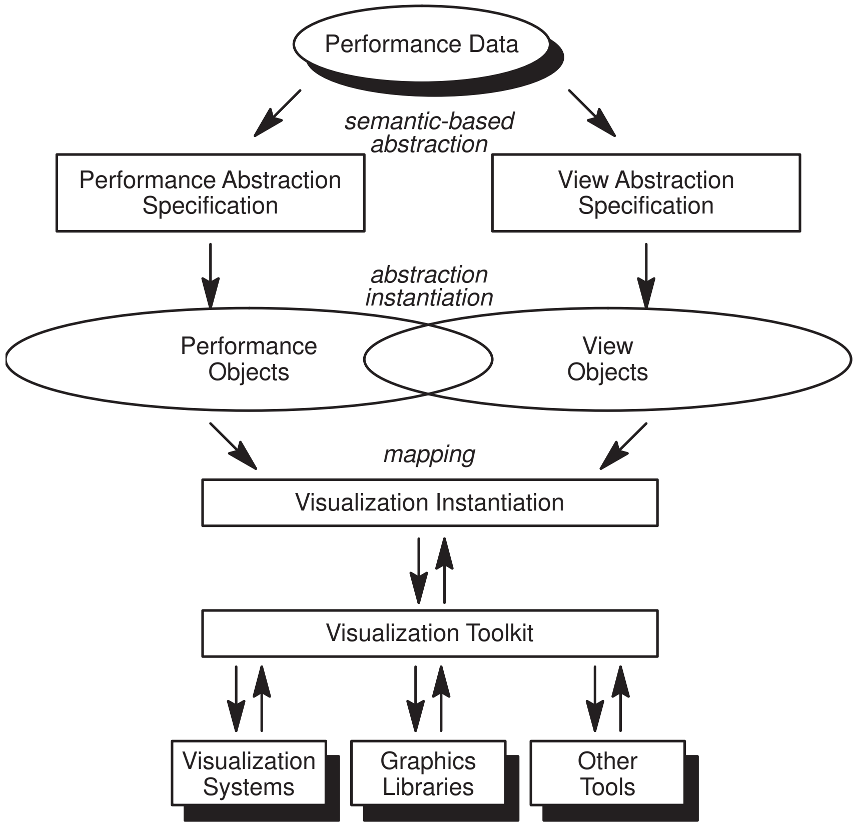
BREEZY
Barrier Breakpoint Debugger

τ

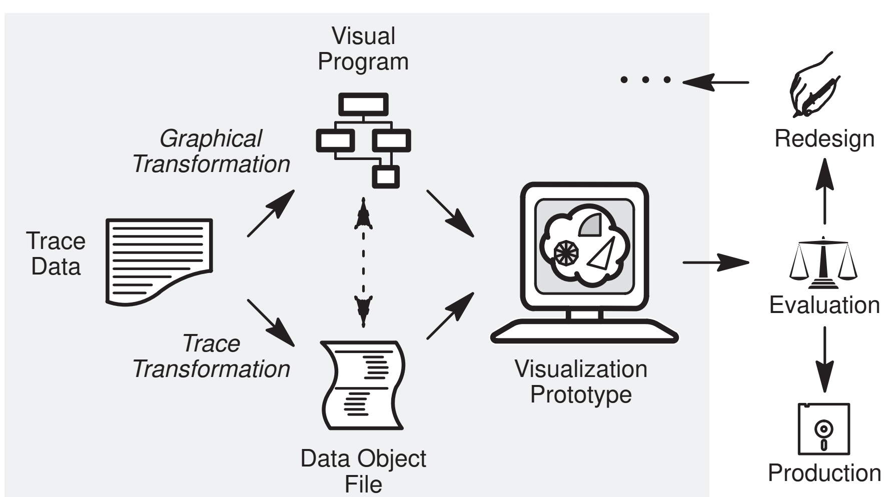


POPEYE
Parallel Program and Performance Displays

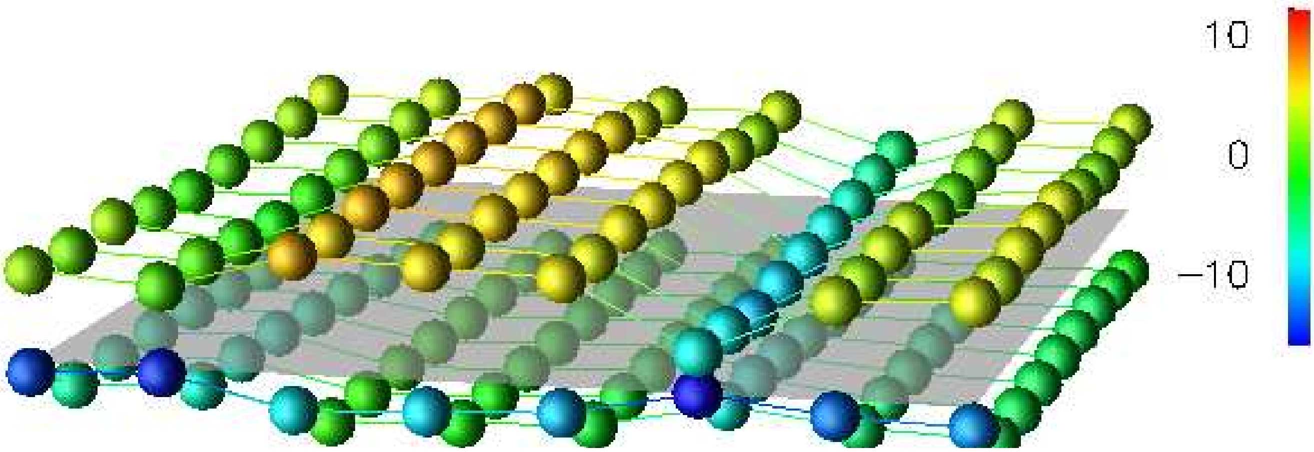
❑ Methodology

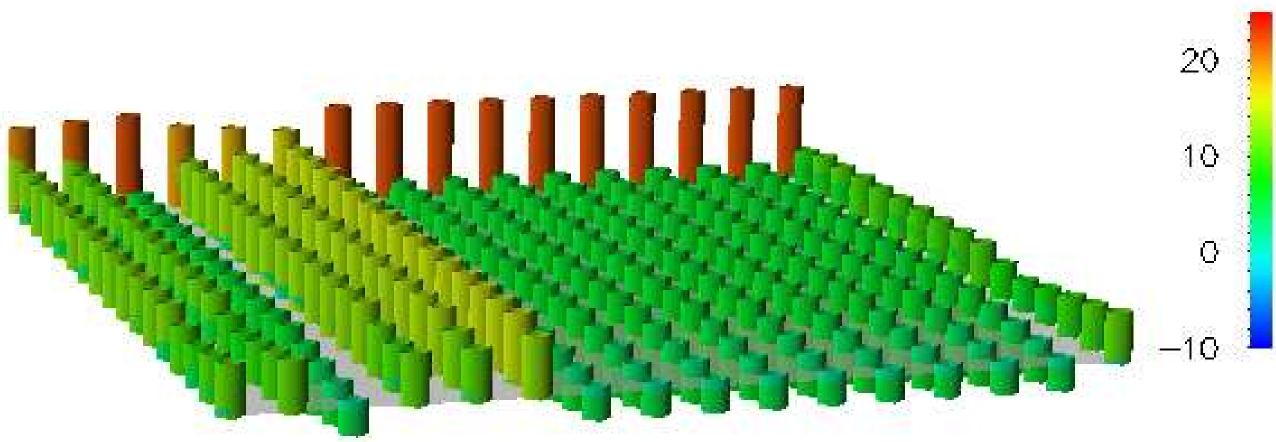


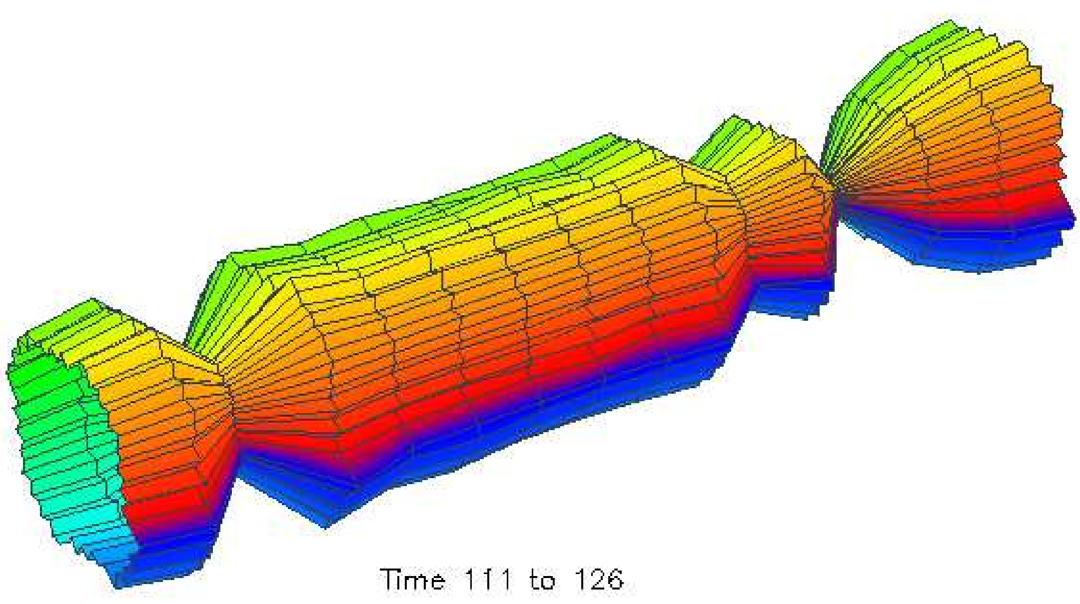
❑ Implementation



POPEYE
Parallel Program and Performance Displays







Related Current Activities

- **The Ariadne/AVE Parallel Debugger**
  (Joydip Kundu / Janice Cuny)
  - Ariadne is a post-mortem, event-based debugger that combines a simple modeling language with functional queries to support thorough exploration of execution traces.
  - Ave is a visualization engine for Ariadne that provides scalable visualizations featuring
    - hierarchical visualizations reflecting the structure of user-defined behavioral models
    - dynamic attribute calculation
    - automatic partitioning of matched behaviors and attributes

- **Poirot: Toward a formal theory of parallel performance diagnosis processes** (Rob Helm)
  - Research goal: a formal theory that:
    - describes how expert parallel programmers diagnose performance problems.
    - explains how performance tools support or inhibit efficient diagnosis.
    - predicts how diagnosis processes will perform in a particular context.
  - Current research includes:
    - analysis of parallel program development histories using the current theory.
    - formalization of the theory in a performance diagnosis toolkit, Poirot, for pC++.

Related Current Activities

- **XtraP: A Performance Extrapolation Tool**
  (Kesavan Shanmugam)
  - uses trace driven simulation to predict the execution time of a pC++ program on a N-node distributed memory machine
  - uses trace files from a N-thread pC++ program running on a uniprocessor machine
  - uses high level events and avoids detailed low-level simulation

- **VL / IVA: Visualization Tools**
  (Harold Hersey / Chris Hundhausen
  - Goal: Develop visualization technology and artifacts that
    - empower scientists to construct their own visualizations of their phenomena of interest; and
    - serve as mediational resources in the intricate processes by which scientists explore and make sense of those phenomena.
  - Current work includes:
    - studies and analyses of visualization scenarios;
    - development of VL, a tree manipulation Visualization Language based on Open Inventor; and
    - development of IVA, an Interactive Visualization Artifact based on VL,TCL, and Fresco.

τ

☆ Tools Are Us ☆

For more Information

- **WWW/Mosaic**:
  - http://www.cs.uoregon.edu/paracomp/tau
  - http://www.cica.indiana.edu/sage
- **Anonymous FTP**:
  - cica.cica.indiana.edu:/pub/sage (129.79.20.27)
- **finger**:
  - sage@cica.indiana.edu
- **mail**:
  - sage@cica.indiana.edu

PARA  DUCKS