UNIVERSITY OF WISCONSIN MADISON

Paradyn **v2.1** Release

Brian J. N. Wylie

`wylie@cs.wisc.edu`

Computer Sciences Department

University of Wisconsin

1210 W. Dayton St.

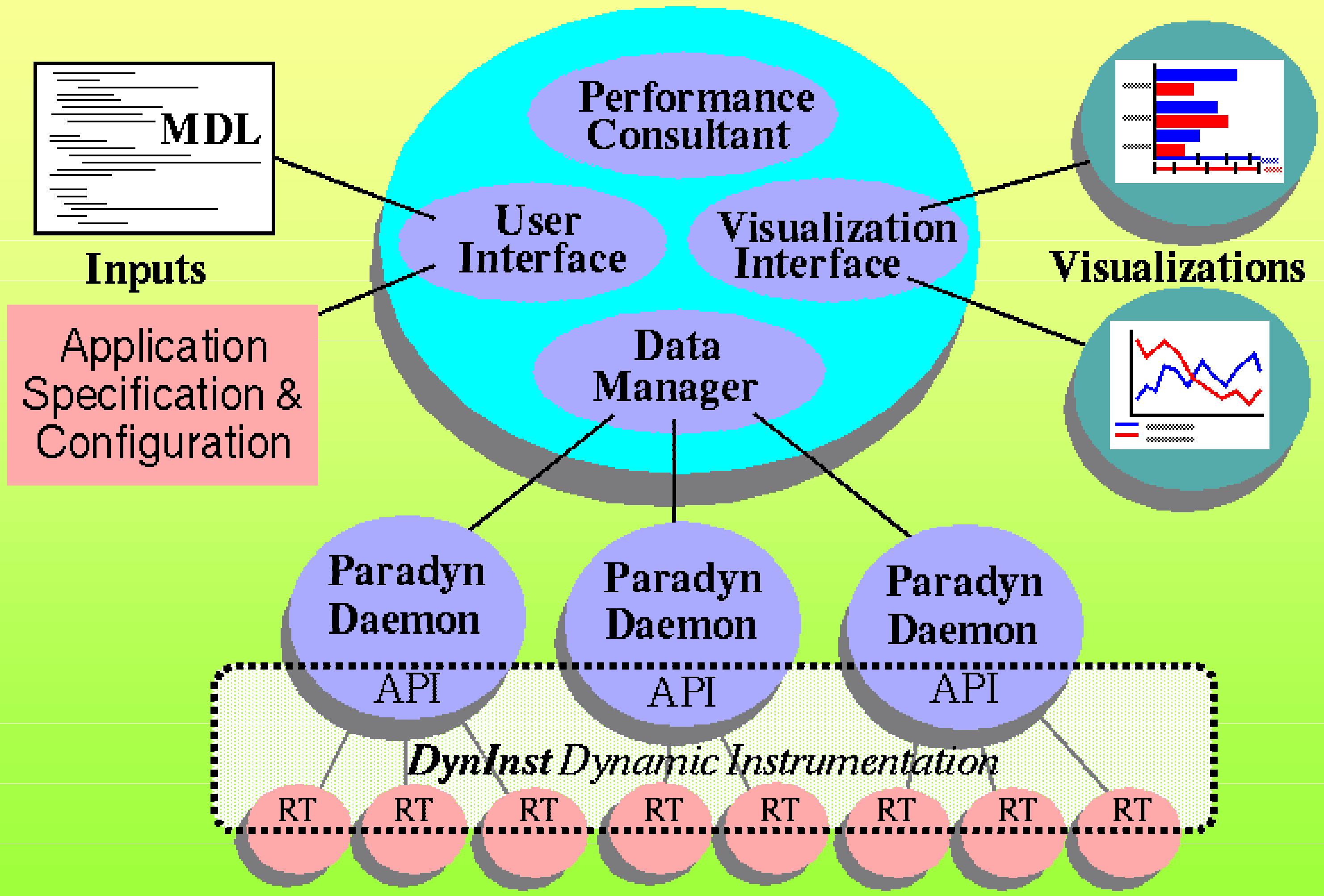
Madison, WI 53706-1685

USA

Outline

- Review of *Paradyn* **v2.0** (Sept.'97)
  - Synchronized *DynInstAPI* **v1.0** release (U. Maryland)
- Developments since **v2.0**
  - Extended capabilities
  - Performance enhancements
  - Generic maintenance
  - Miscellaneous bug-fixes
- Current status

Paradyn & *DynInst* Architecture



Application Processes

Summary of *Paradyn* **v2.0**

- Key features:
  - Basic support for MPI [under POE on SP2-AIX]
  - New x86-WindowsNT platform [*paradynd*]
  - Dynamic linking of *libdyninstRT* [SPARC-Solaris]
  - No re-linking requirement [x86-WindowsNT]
- Released Sept.'97 (sources & binaries)
  - Synchronized initial *DynInstAPI* **v1.0** release
  - Occasional subsequent interim releases

Paradyn **v2** functionality summary

Key:

♥	Support currently under development
♣	Applications compiled by VC++ only
♦	Support added in *DynInstAPI* **v1.1** only
♠	Programs started under SP2 POE only

	SPARC Solaris	**x86 Solaris**	**x86 WinNT**	**RS6000 AIX**
Front-end/GUI (*paradyn* & *Visis*)	✔	✔	✘	✔
Daemon (*paradynd* & *libdyninstRT*)	✔	✔	✔ ♣	✔
***DynInstAPI* library**	✔	✔	✔ ~~♣~~ 2.1	✔
Shared-objects / dynamic linking	✔	✔	✔	✘
***libdyninstRT* as a shared library**	✔	✘ ➡ 2.1 ✔	✔	✘
Dynamic loading of *libdyninstRT*	✘ ➡ 2.1 ✔	✘ ♥	✔	✘
Attach to running process(es)	✔	✔	✔	✘ ♦
Supported parallel execution modes	**PVM**	**PVM**	**PVM**	**PVM** **MPI**♠

Example of "linking" revisions

Makefile:

```
  SYSLIBS  = -lm -lsocket -lnsl
  OBJECTS  = main.o this.o that.o
- PDOBJECTDIR=$PARADYN_ROOT/lib/$PLATFORM
~ PARADYN_LIB=$PARADYN_ROOT/lib/$PLATFORM/libdyninstRT.so
  app:    $(OBJECTS)
          $(CC) -g -o app \
-               $(PDOBJDIR)/DYNINSTstartCode.o \
                $(OBJECTS) \
-               $(PDOBJDIR)/DYNINSTendCode.o \
-               $(PARADYN_LIB) \
                liblots_of_stuff.a $(SYSLIBS)
```

paradyn.rc or **app.pcl**:

```
  exclude "/Code/libc.so.1";              // 1000's of fns
+ exclude "/Code/liblots_of_stuff.a";     // uninteresting
  ...
```

v2.1 extended capabilities (&)

- Automatic code block identification [Solaris]
  - eliminates requirement for application re-linking using explicit dyninstSTART/ENDcode markers
  - exclusion of statically-linked modules & functions
- 2-pass function re-locator/expander [SPARC/Solaris]
  - undoes (some) tail-call optimizations to allow full instrumentation of highly-optimized functions
- Handling stripped *dynamic* libraries [Solaris]
  - use run-time linker's dynamic symbol table (.dynsym)

v2.1 extended capabilities (& cont.)

- Robust handling of larger processor sets
  - Multiple retries of *paradynd* connections
- Handling multiple *paradynd*s per processor

DynInstAPI **v1.1** only:

- Blocking option to wait for any events
- Parsing gcc-compiled executables [x86-WNT]

v2.1 extended capabilities (UI)

- More powerful MDL syntax
- Metrics for I/O in MPI programs [SP2-AIX]
- External *paradynd* start-up support
  - `UW-SP2> paradyn -f app.pcl -x ~/.paradynd`
  - `$ paradynd -z<flavor> -l2 -mUW-SP2 -p12345`
- Refined user interface
  - Scalable process status area (with scrollbar!)
  - Distinct information & error message displays
  - Handling goofy characters in function identifiers

MDL: Metric Description Language

Specification of instrumentation operations which can be applied by paradynd to application instrumentation points

```
if (<metric_expr>) {                              // v2.0
    foreach func in <metric_expr>
        (* if (<inst_expr>) <inst_request_expr>; *)
}
```

- More powerful, consistent expression syntax

```
if (<expr>) {                                     // v2.1+
    foreach func in <expr>
        (* if (<expr>) <expr>; *)
}
```

  - any valid expression now acceptable as function arguments
  - replace/update **`paradyn.rc`** configuration files!

Updated MDL expression syntax

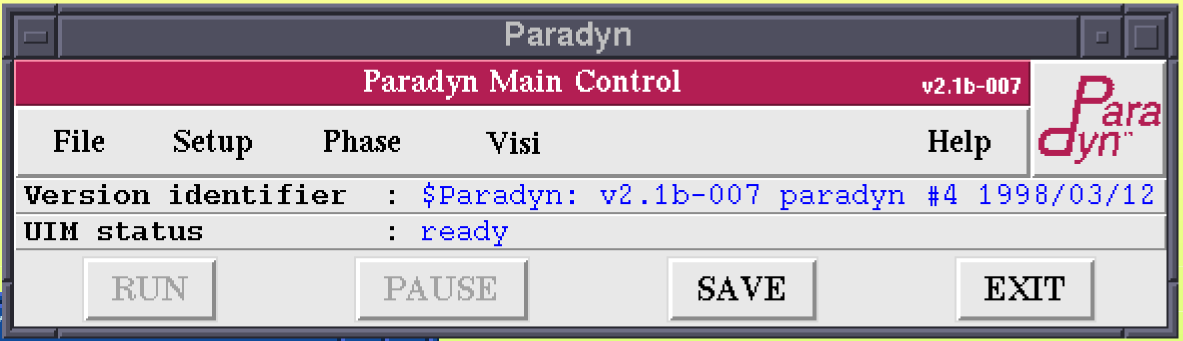
Former: Updated:

- `setCounter(i,j)` → `i=j`
- `addCounter(i,j)` → `i=i+j | i+=j` // `i++`
- `subCounter(i,j)` → `i=i-j | i-=j` // `i--`

New from **v2.1**:

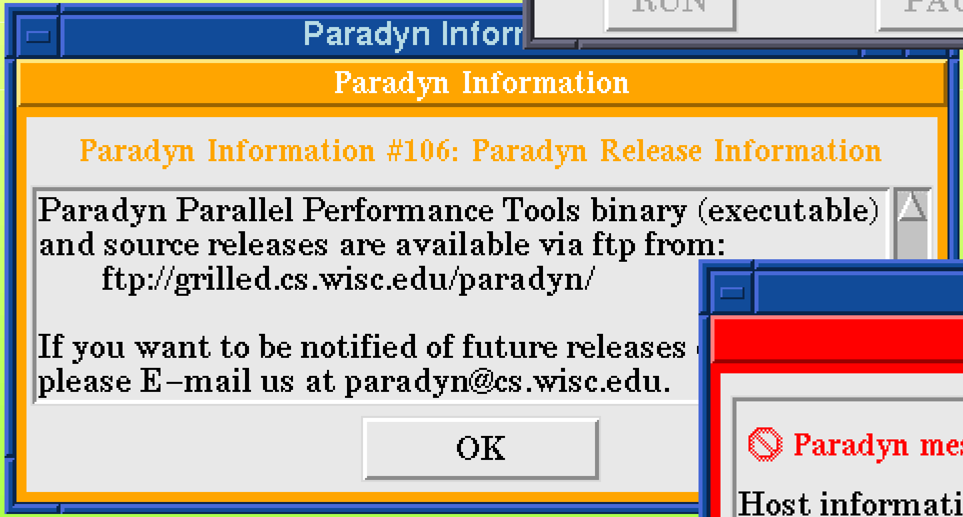
```
foreach callsite in func.calls {
        append preInsn callsite
    (* // any valid expression can be function args
        if ($arg[n1*(n2+n3)] == "xyz")
            counter = Func1(-(a*b),c) + Func2(d,e,f);
    *)
}
```

Paradyn v2.1 error & info displays



```
Version identifier  : $Paradyn: v2.1b-007 paradyn #4 1998/03/12
UIM status          : ready
```

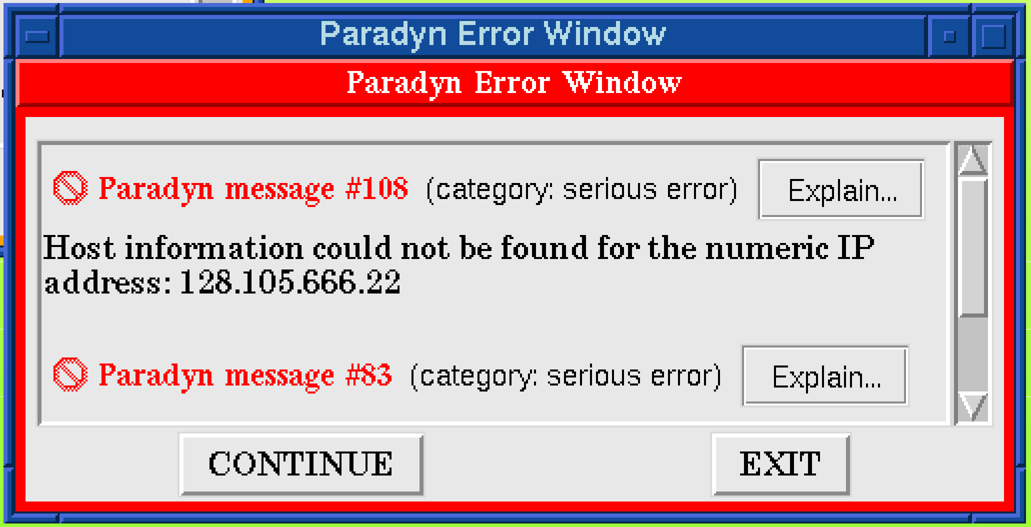
•Basic help menu



Paradyn Information #106: Paradyn Release Information

Paradyn Parallel Performance Tools binary (executable)
and source releases are available via ftp from:
ftp://grilled.cs.wisc.edu/paradyn/

If you want to be notified of future releases
please E-mail us at paradyn@cs.wisc.edu.



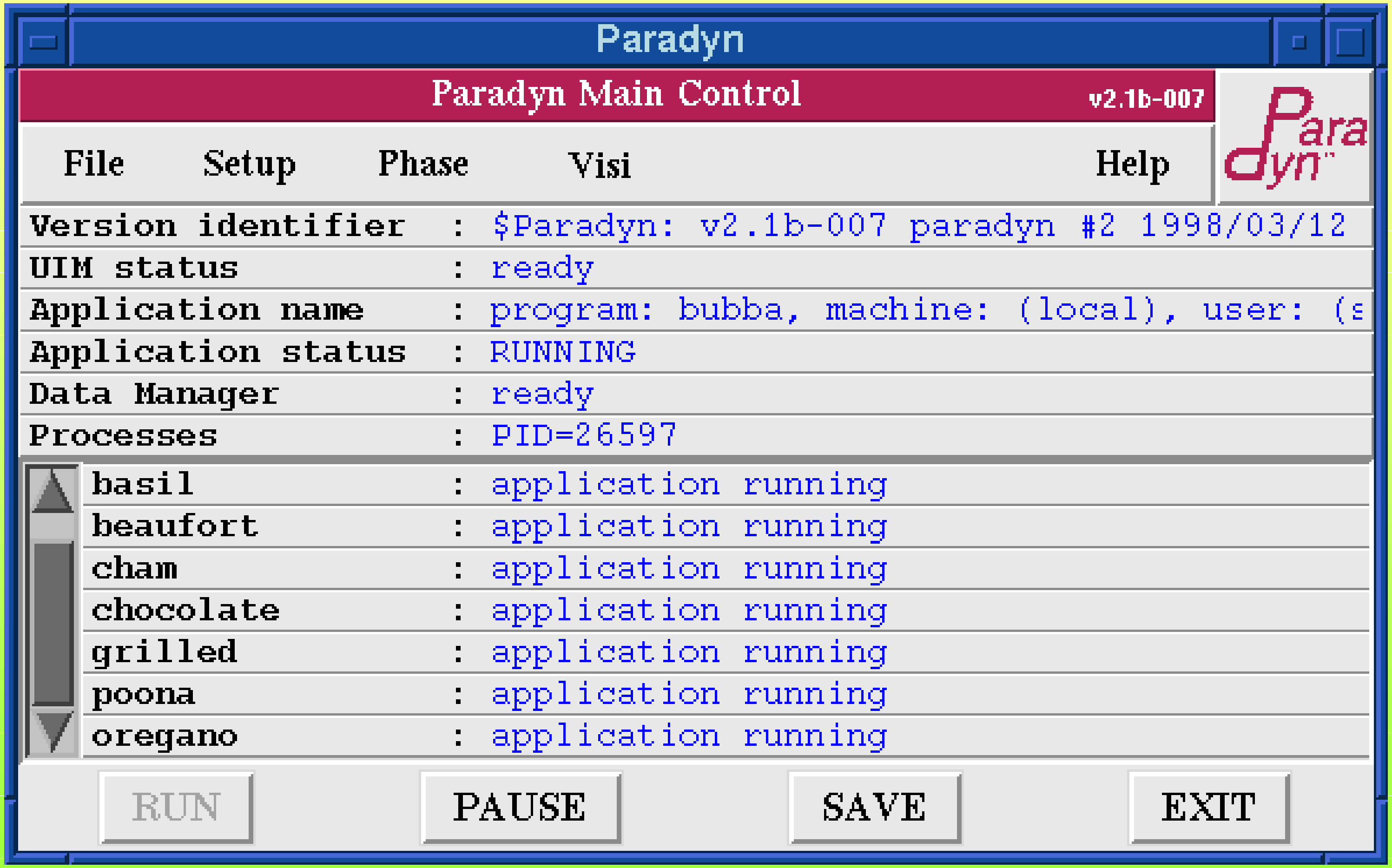
Paradyn message #108 (category: serious error)

Host information could not be found for the numeric IP address: 128.105.666.22

Paradyn message #83 (category: serious error)

•New non-modal information display

Paradyn **v2.1** Main Control window



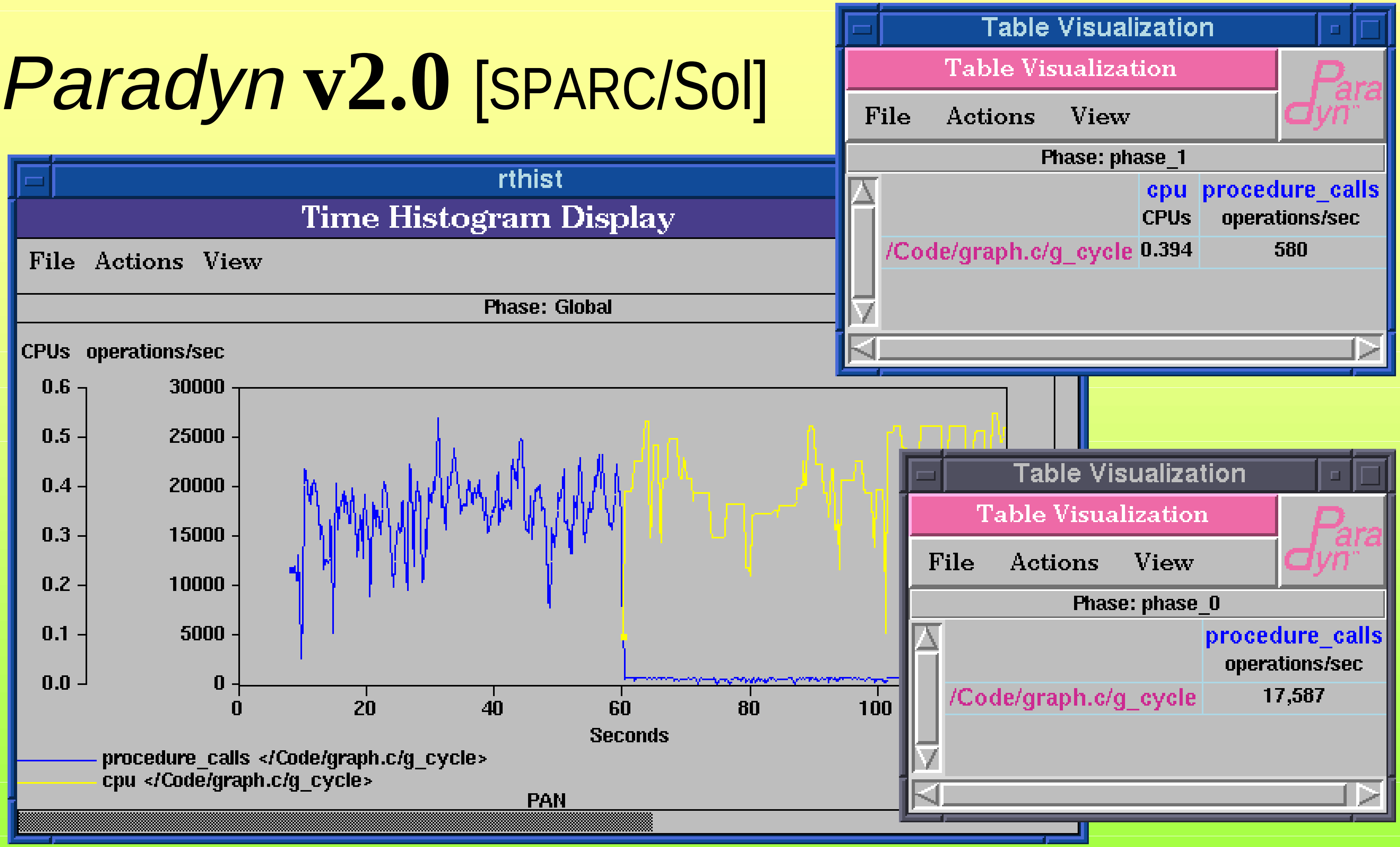
```
Version identifier  : $Paradyn: v2.1b-007 paradyn #2 1998/03/12
UIM status          : ready
Application name    : program: bubba, machine: (local), user: (s
Application status  : RUNNING
Data Manager        : ready
Processes           : PID=26597
basil               : application running
beaufort            : application running
cham                : application running
chocolate           : application running
grilled             : application running
poona               : application running
oregano             : application running
```

•Separate resizable, scrollable area for process status info.

v2.1 performance enhancements

- Disclaimer: ***your actual mileage will vary!***
- Faster location of program symbols (typ. ×2)
  - part of executable parsing on start-up/attach
  - better handling of large numbers of functions in complex applications (& non-excluded libraries)
- Optimized instrumentation [x86-Solaris]
  - avoiding some of the use of costly traps ✥ *Buck*
- Switch to lower-overhead system timer [Solaris]
  - ***/proc*** PIOCUSAGE → gethrvtime() : 64µs → 2µs!

Paradyn **v2.0** [SPARC/Sol]

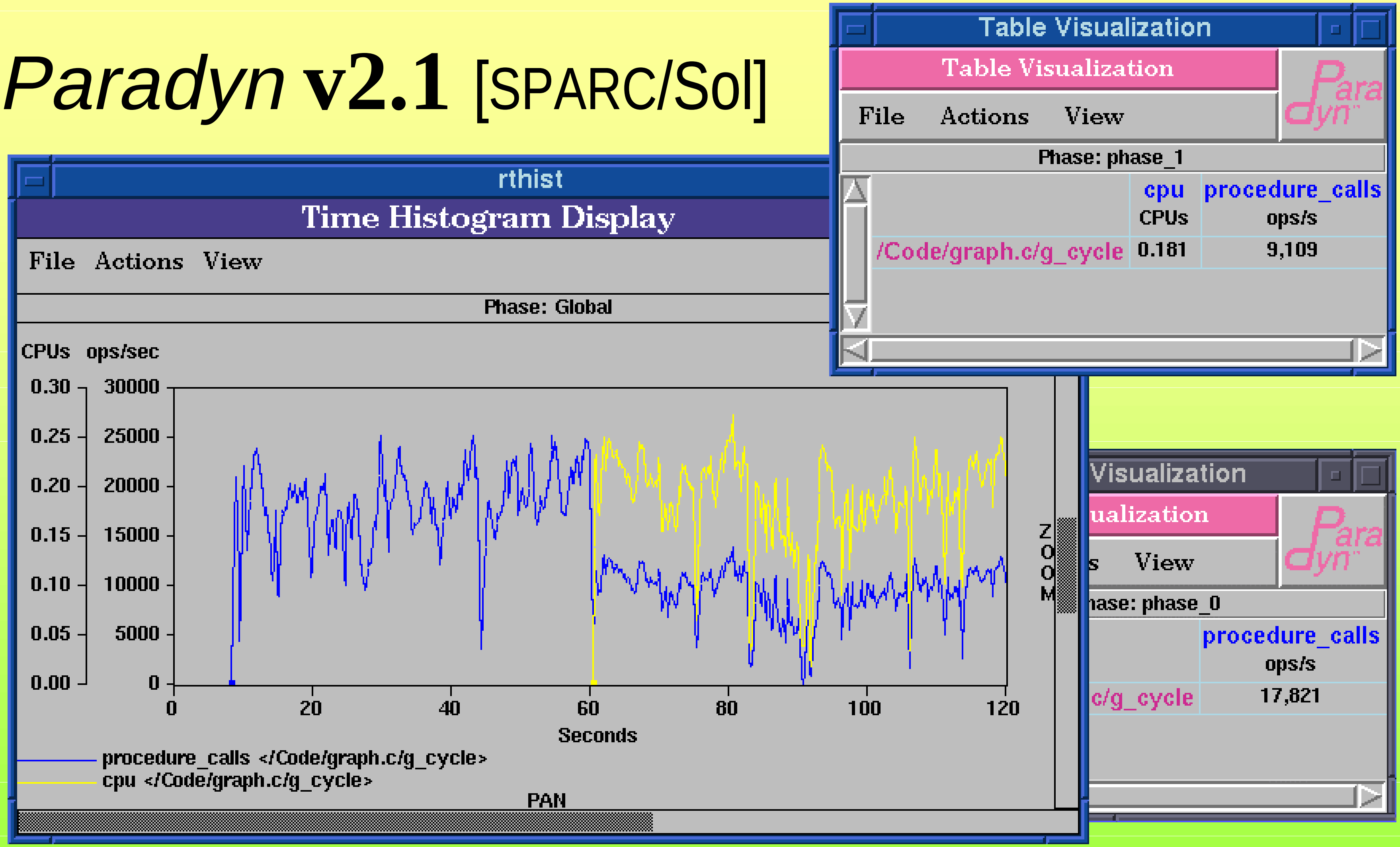


	cpu CPUs	procedure_calls operations/sec
/Code/graph.c/g_cycle	0.394	580

	procedure_calls operations/sec
/Code/graph.c/g_cycle	17,587

CPU metric added (to procedure_calls base instrumentation) after 60s reduces effective performance *over* ***30****-fold!*

Paradyn v2.1 [SPARC/Sol]



Phase: phase_1

	cpu CPUs	procedure_calls ops/s
/Code/graph.c/g_cycle	0.181	9,109

hase: phase_0

	procedure_calls ops/s
c/g_cycle	17,821

Less intrusive timers only reduce performance around 50%, providing more accurate measurements (*c.f.* CPU in phase 1)

v2.1 performance enhancements (cont.)

- *Paradyn* used to analyze its own performance!
  - *Paradyn*/*paradynd* are analyzing a subject application
    - the *Perf. Consultant* is conducting an automated search
  - A 2nd *Paradyn*/*paradynd* is attached to the 1st *paradynd*
    - *Naím*'s expert analysis of performance of 1st *paradynd* identifies excessive pausing & continuing of application, **but** this is done in many different places inside *paradynd* !
    - *Figueira*'s prototype path-profiling tool isolates excesses to *paradynd* 's method of modifying subject instrumentation
- Faster instrumentation enabling/disabling (×16)
  - minimize *paradynd* interference with running processes

v2.1 generic software maintenance

- Easier source build strategy
  - Configurable from top-level **`Makefile`**s
  - Optional build to incorporate support for PVM
- Integrated build identification information
- Tcl/Tk upgrade to v8.0
  - X→Tk portable font substitution
- General source tidy and reduction of the number of warnings during compilation!

Elimination of key bugs in **v2.0**

- *Paradyn* front-end data-collection memory leak
- Handling pending system calls when application paused
- Improper *Visi* trace-stream closing
- *Igen* parsing of (invalid/incomplete) argument lists
- Race condition in *paradynd* main control loop
- ...

Fixes for *DynInstAPI* **v1.0**:

- Buffer mis-alignment to word boundaries [SPARC-Solaris]
- Improved parsing of Portable Executable format images and jump-tables [x86-WNT]

Beyond **v2.1** – the near future

- Support for multithreaded applications ✥ *Naím*
- Improved *Performance Consultant* search ✥ *Cheyney*
- Ports for DEC-Alpha, x86-Linux, MIPS-IRIX
- Handling machines/hosts specified by numeric IP addresses
- Dynamic loading of *libdyninstRT* on x86-Solaris
- Handling relocated dynamically-loaded libraries on x86-WinNT
- Guaranteed instrumentation of program `main()`
- Source code profile viewer & code-coverage *Visi*
- Portable (entirely) Tcl/Tk-based GUI
- Clean detach from attached application processes
- ... other exotica ...

Current status

- Release targeted for end of March '98
  - code freeze in effect — no new functionality
  - extensive testing in progress
  - documentation synchronization check
  - both source & binary packages will be available:
    `ftp://grilled.cs.wisc.edu/paradyn/`
    `http://www.cs.umd.edu/~hollings/dyninstAPI/`
- Provide your feedback (`paradyn@cs.wisc.edu`) about experiences, difficulties, priorities & requirements to guide us improving *Paradyn* & *DynInstAPI*