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

Inputs
- Application Specification & Configuration

MDL

Performance Consultant
- User Interface
- Visualization Interface
- Data Manager

Visualizations

Paradyn Daemon
- API
- Dynamic Instrumentation

DynInst

Application Processes
- RT RT RT RT RT RT RT RT
## 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

<table>
<thead>
<tr>
<th>Functionality</th>
<th>SPARC Solaris</th>
<th>x86 Solaris</th>
<th>x86 WinNT</th>
<th>RS6000 AIX</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Front-end/GUI (paradyn &amp; Visis)</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Daemon (paradynd &amp; libdyninstRT)</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>DynInstAPI library</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td><strong>Shared-objects / dynamic linking</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td><strong>libdyninstRT as a shared library</strong></td>
<td>✔️</td>
<td>✗</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Dynamic loading of libdyninstRT</strong></td>
<td>✗</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Attach to running process(es)</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✗</td>
</tr>
<tr>
<td><strong>Supported parallel execution modes</strong></td>
<td>PVM</td>
<td>PVM</td>
<td>PVM</td>
<td>PVM MPI†</td>
</tr>
</tbody>
</table>

**Key:**
- ✔️: Support currently under development
- ♣: Applications compiled by VC++ only
- ♦: Support added in DynInstAPI v1.1 only
- ♤: Programs started under SP2 POE only

© 1998 wylie@cs.wisc.edu
Example of "linking" revisions

**Makefile:**

```bash
SYSLIBS  = -lm -lsocket -lnsl
OBJECTS  = main.o this.o that.o
- PDPRODUCTDIR=$PARADYN_ROOT/lib/$PLATFORM
- PARADYN_LIB=$PARADYN_ROOT/lib/$PLATFORM/libdyninstRT.so
app: $(OBJECTS)
   $(CC) -g -o app \\
   $(PDPRODUCTDIR)/DYINSTStartCode.o \\
   $(OBJECTS) \\
- $(PDPRODUCTDIR)/DYINSTEndCode.o \\
- $(PARADYN_LIB) \\
   liblots_of_stuff.a $(SYSLIBS)
```

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

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

© 1998 wylie@cs.wisc.edu    [6]  Paradyn v2.1 Release
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 paradynds 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> paradynd -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

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

- More powerful, consistent expression syntax

```c
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:
- `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:

```java
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

- New non-modal information display
- Basic help menu
Paradyn v2.1 Main Control window

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

• 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