Performance Tools for Parallel and Grid Computing

Daniel Reed

Celso Mendes, Dan Wells, Ying Zhang

Pablo Research Group

http://www-pablo.cs.uiuc.edu

Performance Expedition – May 29, 2003

Presentation Outline

- Performance Analysis on Parallel Systems
  - ⇒ SvPablo Performance Browser
  - ⇒ Steps for Instrumentation and Analysis
- Performance Analysis on the Grid
  - ⇒ Autopilot Toolkit
- I/O Performance Analysis
  - ⇒ Pablo I/O Tools (PCF)

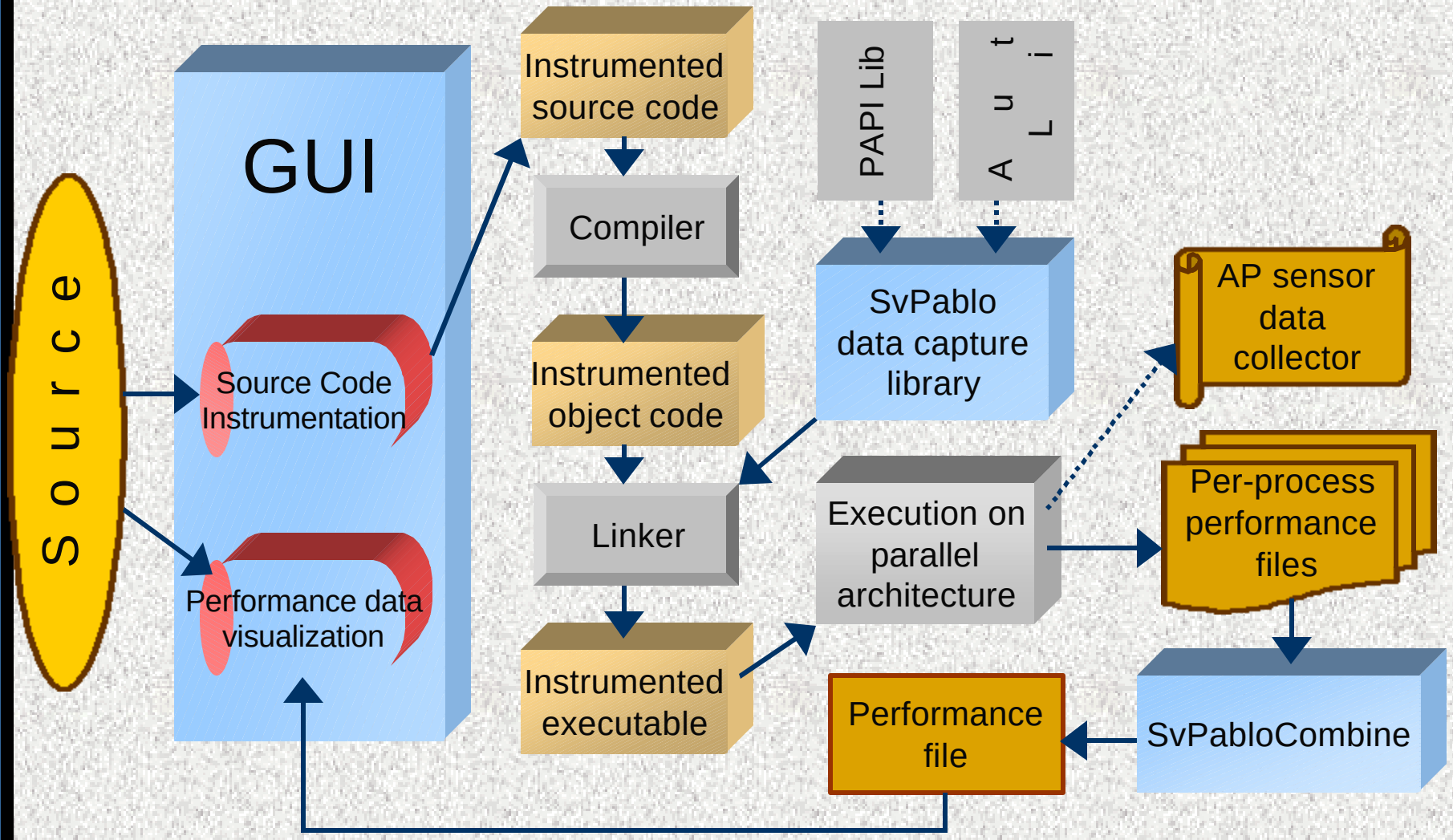
SvPablo Performance Browser

- Graphical performance analysis environment
  - source code instrumentation
  - performance data capture, browsing and analysis
  - F77/F90 and C language support
- Performance capture features
  - software/hardware performance data
    - loop and procedure counts/durations
    - hardware performance counter data, via PAPI
  - statistical summaries for long-running codes
  - option for real-time data transmission via Autopilot
- Availability
  - source code, binaries, dataset examples
  - platforms: Sun Solaris, IBM SP, SGI Origin, Linux IA-32/IA-64, Compaq Alpha, NEC SX-6, Sony PlayStation-2

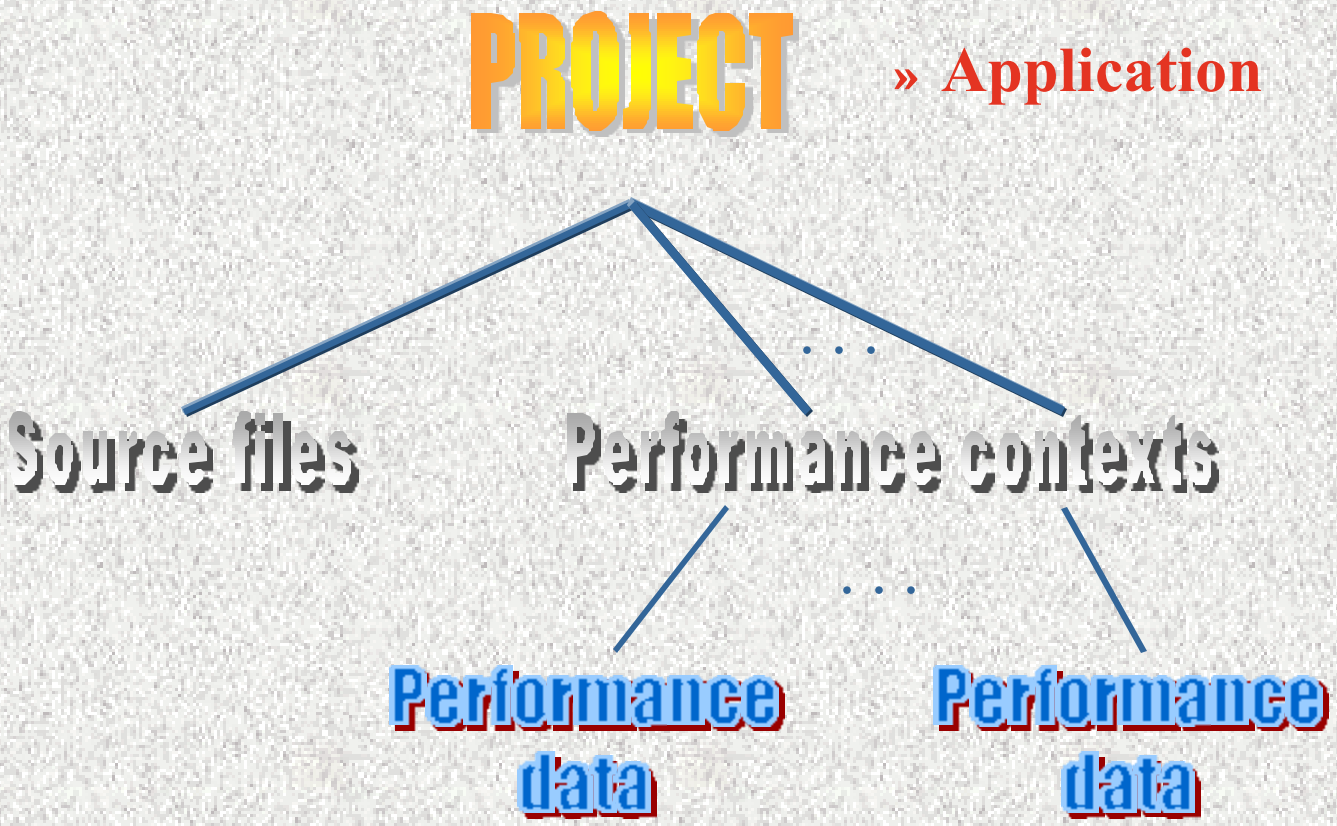
http://www-pablo.cs.uiuc.edu/Software/SvPablo/svPablo.htm

Platinum:/usr/apps/tools/SvPablo Titan:/usr/apps/tools/Pablo

SvPablo Components



SvPablo Model



Source Code Instrumentation

- Instrumentable constructs
  - procedure calls
  - outer loops
- Basic metrics
  - counts
  - inclusive durations
  - exclusive durations
- Optional metrics
  - any metric provided by PAPI
- Two ways to instrument a code
  - interactive instrumentation (line-by-line)
  - automatic instrumentation: all loops, all procedure calls

Interactive Instrumentation

svPablo

Project Instrument View GenCallGraph Help

Project Description: MILC on IA–64

Source Files:
- control.c
- setup.c
- update.c
- d_congrad5.c
- com_mpi.c

Performance Contexts:
- IA–64 with 4 Processors
- IA–64 with 16 Processors

Routines in Source File
- main
- initialize_machine
- g_sync
- setup
- setup_analyze

Routines in Performance Data

Source File: /u/ncsa/svpablo/Celso/MILC/sources/control.c

```
initialize_machine(argc,argv);
g_sync();
    /* set up */
    prompt = setup();
    setup_analyze();

    /* loop over input sets */
    while( readin(prompt) == 0){
  /* perform warmup trajectories */
dtime = -dclock();
for(traj_done=0; traj_done < warms; traj_done++ ){
            update();
}
if(this_node==0)printf("WARMUPS COMPLETED\n");

/* perform measuring trajectories, reunitarizing and measuring  */
meascount=0;  /* number of measurements   */

plp = cmplx(99.9,99.9);
avm_iters = avs_iters = 0;
```

Instrument/Clear Line View Line Data

Instrumentable lines

Instrumented lines

Click on line to toggle instrumentation on / off

Instrumented Code Execution

- Adjust application's Makefile(s)
  - replace source code filenames
    e.g. prog.c → prog.ContextName.inst.c
  - compile InstrumentationInit.c and link in (automatically created by SvPablo)
  - OBS: must always instrument main program
  - link with SvPabloLibrary (and PAPI-Library)
- Execute instrumented executable
- Collect per-task performance files
  - SvPabloCombine –o PerfFile c_SDDF*.ascii
- Go back to GUI for performance browsing

SvPablo Performance Browsing

svPablo

Project Instrument View GenCallGraph Help

Project Description: MILC on IA-64

Source Files:
control.c
setup.c
update.c
d_congrad5.c
com_mpi.c

Performance Contexts:
MILC on 4 IA-64 Processors
MILC on 8 IA-64 Processors
MILC on 16 IA-64 Processors
Test Context
MILC on 4 IA-64 Processors using PAPI
MILC on 16 IA-64 Processors using PAPI

Routines in Source File
ks_congrad
MILC_dclock
deanup_gather
dslash
scalar_mult_add_su3_vector

Routines in Performance Data
ks_congrad
update
dslash_special
scalar_mult_add_su3_vector
mult_su3_mat_vec_sum_4dir

Source File: /u/nosa/cmendes/Celso/MILC-SC/sources/d_congrad5.c

```
        dir, parity, gen_pt[dir] );
  ~> > else restart_gather( src, sizeof(su3_vector),
        dir,
      |

  #ifdef DS
      dtime
      dtime
  #endif

  #ifdef MYTIME
  dtimem -= dclock();
  #endif
      /* Multiply by adjoint matrix at other sites */
  ~> >  FORSOMEPARITYDOMAIN(i,s,otherparity){
  ~> > mult_adj_su3_mat_vec_4dir( s->link,
        (su3_vector *)F_PT(s,src), s->tempvec );
      | END_LOOP
```

Instrument/Clear Line View Line Data

Specific Metric
Call Statistics
count:
1319328.0000 -- mult_adj_su3_mat_vec_4dir
Dismiss Help

Specific Metric
Call Statistics
Duration:
19.5759 -- mult_adj_su3_mat_vec_4dir
Dismiss Help

Specific Metric
HW Statistics by Line
Floating Point Instructions:
242349871.0000 -- mult_adj_su3_mat_vec_4di
Dismiss Help

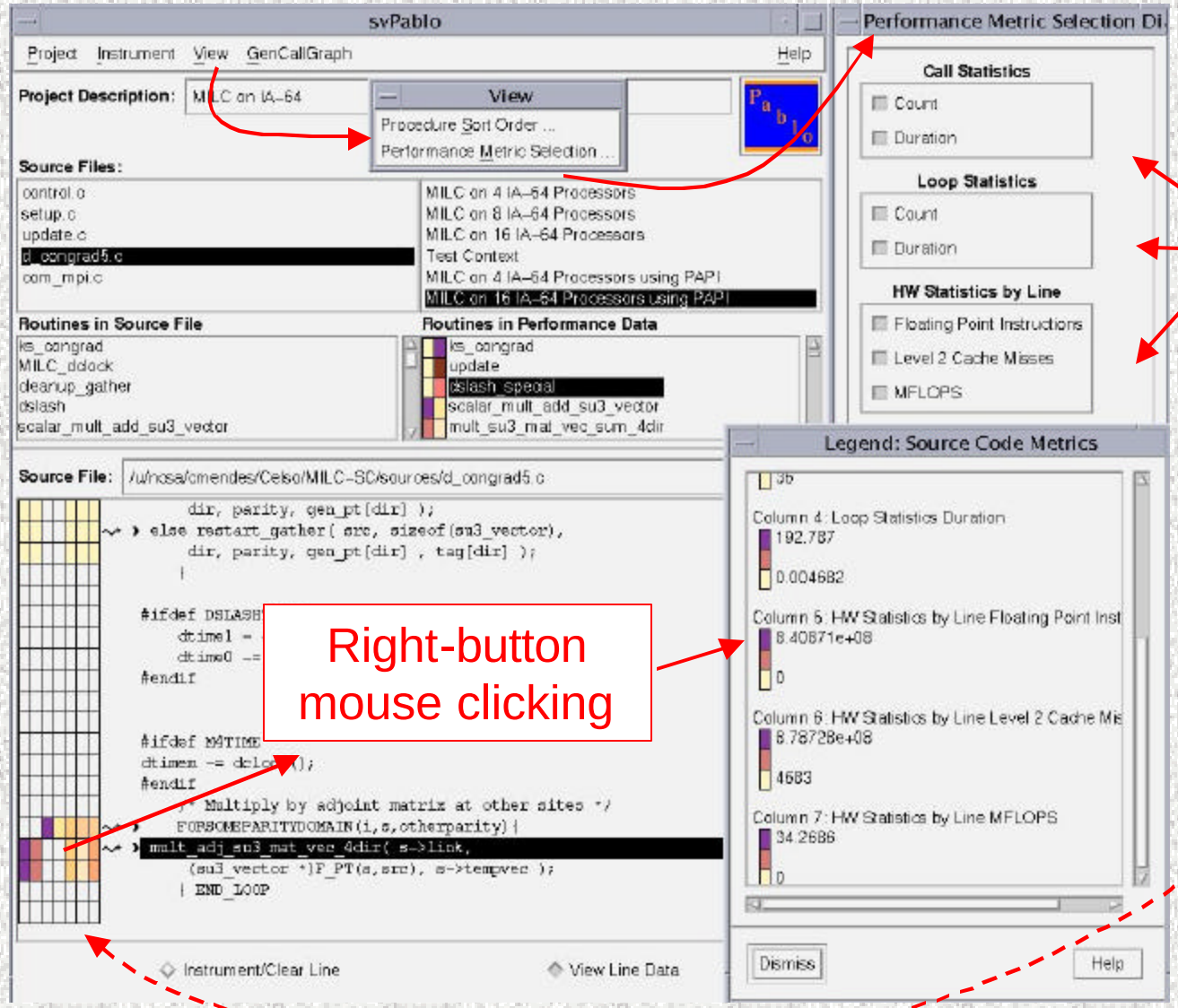
Specific Metric
HW Statistics by Line
Level 2 Cache Misses:
61678226.0000 -- mult_adj_su3_mat_vec_4di

Specific Metric
HW Statistics by Line
MFLOPS:
12.4821 -- mult_adj_su3_mat_vec_4dir
Dismiss Help

Left-button mouse clicking

Captured metrics

SvPablo GUI Configuration



Right-button mouse clicking

Show / hide box columns

Performance Statistics by Line

Performance Data: Call Statistics // mult_adj_su3_mat_vec_4dir // mult_adj_su3_mat_vec_4dir

Tasks: 0 .. 15

File Name(s): d_congrad5.c

Routine Name:

Line Number: 574

Source Code Fragment:

```
        mult_adj_su3_mat_vec_4dir( s->link,
            (su3_vector *)F_PT(s,src), s->tempvec );
    } END_LOOP

#ifdef M4TIME
```

Across tasks

Field Name	Mean	Max Value	Max Task	Min Value	Min Task	Std Dev
Count	1319328.000000	1319328.000000	0	1319328.000000	0	0.00000000
Seconds	19.479348	19.575926	8	19.399450	6	0.04882241
Exclusive Seconds	19.479348	19.575926	8	19.399450	6	0.04882241
FP Instructions	242235001.062500	242349971.000000	13	241916430.000000	11	106850.60419109
L2 Cache Misses	57815837.375000	61678226.000000	15	53529952.000000	13	2253485.70302831

View detailed performance data

OK

Help

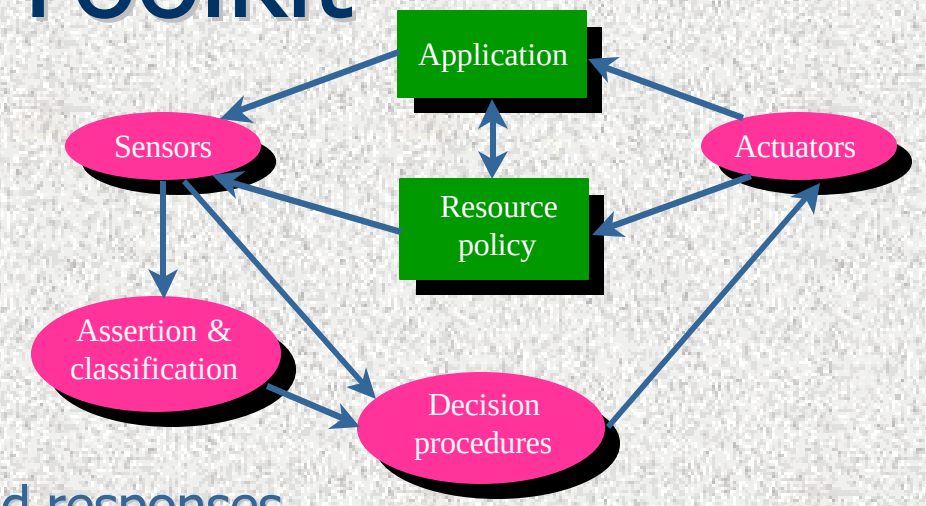
Captured metrics

Request more details…

SvPablo Application Examples

- Cactus
  - insfrastructure for astrophysics simulations
  - tests with Fortran77 thorns (SC'99 demo, with Autopilot)
  - platform for instrumentation: SGI Origin
- CSAR Codes
  - physics in rocket simulations
  - mix of C, C++ and Fortran90
  - platforms for instrumentation: Linux IA-32, IBM-SP
- MILC Code
  - QCD / lattice gauge simulations
  - main code in C, some libraries in assembly (SC'01 demo)
  - platforms for instrumentation: Linux IA-32 & IA-64
- PCTM, POP Codes
  - numerical weather simulation (POP: ocean simulation)
  - most of the code in Fortran90
  - PCTM: more than 400 source files, ~ 60 modules
  - platform for instrumentation: IBM-SP

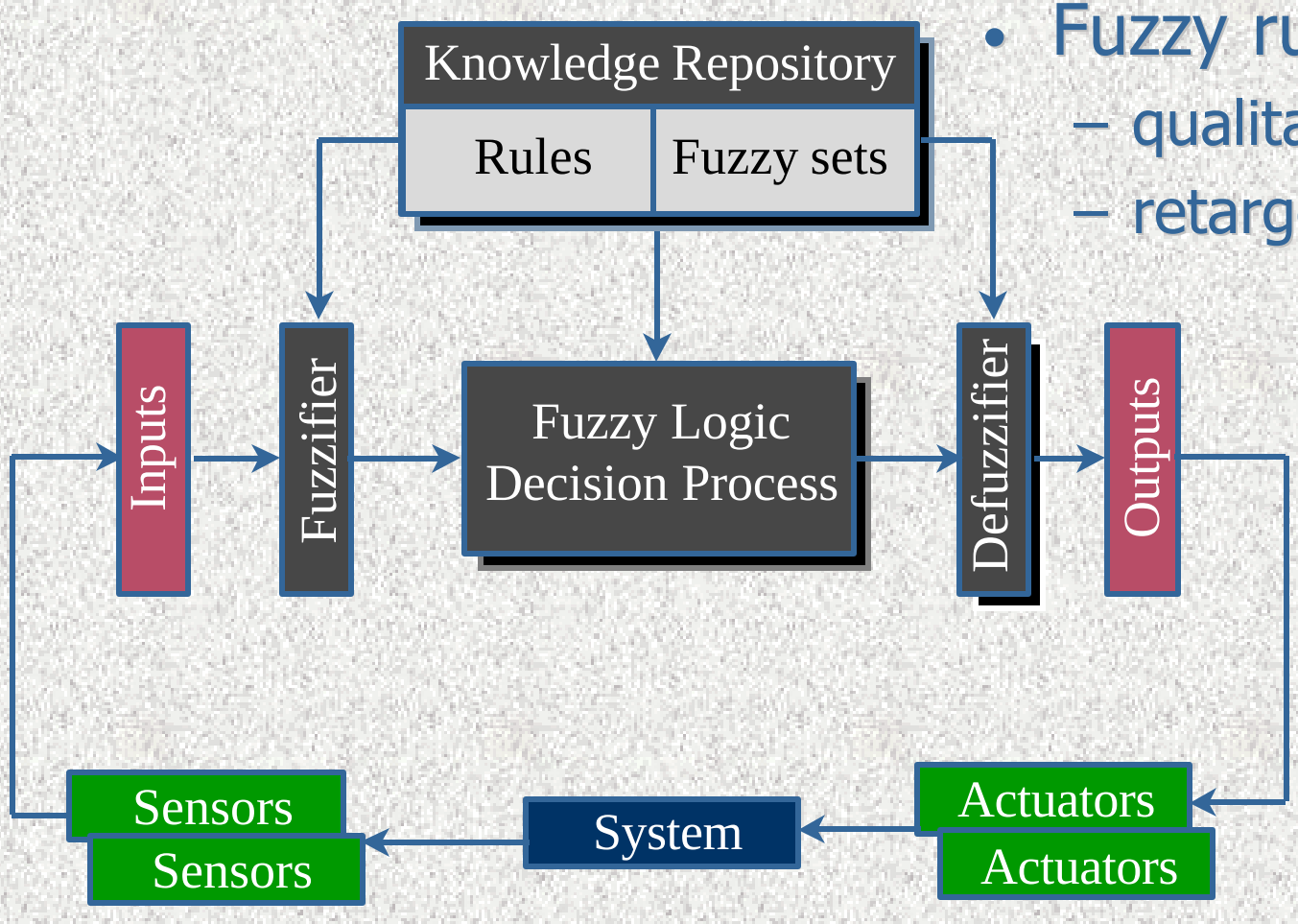
Autopilot Toolkit



- Rationale
  - adaptive applications
  - dynamic resource demands
  - closed loop control
- Research approach
  - monitor resource demands and responses
  - select policies based on observed behavior
  - implement policy changes locally and globally
  - support heterogeneous, distributed executions (via Globus)
- Availability
  - source code, support utilities
  - platforms: Solaris, IBM-SP, SGI Origin, Linux IA-32/IA-64

http://www-pablo.cs.uiuc.edu/Software/Autopilot/autopilot.htm

Autopilot Decision Process



- Fuzzy rule base
  - qualitative behaviors
  - retargetable

Pablo I/O Analysis Tools

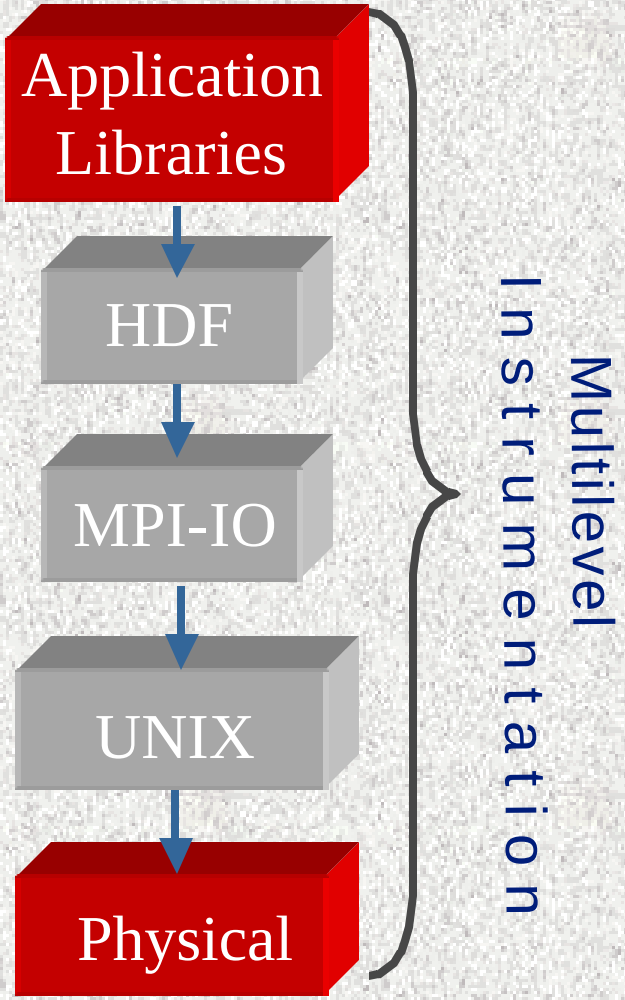
General I/O Analysis Scenario

- Application uses a library with mix of HDF, MPI I/O and Unix I/O calls
- With Pablo tools, each layer can be instrumented for calls to lower layers (even physical I/O on Linux)

Implementation

- Performance Capture Facility (PCF)
- Tools for instrumentation and analysis
- Interface to Autopilot sensors

Availability

- Source code, binaries, tutorials

www-pablo.cs.uiuc.edu/Project/CADRE/index.htm