

# SvPablo: A multi-language Architecture-Independent Performance Analysis System

Luiz A. De Rose and Daniel A. Reed

Presented by Jusub Kim

11/08/05 CMSC714



## Goal of their work.

- Development of a GUI tool kit which can...
  - Correlate sequential/ parallel source code with dynamic performance data from both software and hardware measurements
  - Provide a portable, intuitive and easily used interface.





## Their achievements.

- SvPablo(source view pablo):
  - Is a **graphical environment** for instrumenting application source code and browsing dynamic performance data.
  - Is **portable, language independent, and performance metric independent**( enabled by using the SDDF-self describing data format ).
  - Supports for applications executing on both **sequential and parallel systems** and **exploits hardware support** of performance counters.



## Details: How did they achieve that?

- **Performance Instrumentation**
  - Hardware performance integration
  - Performance Visualization
  - Language and Architecture Transparency

## Performance instrumentation



- Supports both interactive and automatic instrumentation.
- **Automatic instrumentation**
  - Support analysis of **HPF** code.
  - Why not allow interactive instrumentation?
    - HPF compiler does high-level optimizations and allowing users to instrument data parallel source code can potentially inhibit any of these optimizations, reducing performance.
  - **Let HPF compiler emit instrumented code**, which calls the svPablo library. It captures data for each executable line and every procedure call.

## Performance instrumentation



- **Interactive Instrumentation**
  - Most compilers for sequential languages primarily focus on local optimizations.
  - Support interactive instrumentation of **C and Fortran**.
  - Restrict instrumentable constructs to **outer loops and procedure calls**.

## Details: How did they achieve that?



- Performance Instrumentation
- **Hardware performance integration**
- Performance Visualization
- Language and Architecture Transparency

## Hardware performance integration



- **Software instrumentaion is simply not enough!** We need to consider also...
  - Superscalar instruction execution effect
  - Cache effect in hardware-managed distributed shared memory hierarchy
- **New microprocessors commonly provide a set of performance registers.** (cycles, cache misses, floating point instructions, branch misprediction, etc.)

## Hardware performance integration



- During program execution, the SvPablo library **queries the user-selected counters** and records them.
- After program execution, the SvPablo **records its statistical analyses in a set of summary files**, one for each executing process and **merges them computing new global statistics**, which is input to the analysis graphical interface.

## Details: How did they achieve that?



- Performance Instrumentation
- Hardware performance integration
- **Performance Visualization**
- Language and Architecture Transparency

Project: Instrument View Help

Project Description: Red Black SOR in C using MPI

Source Files: prbsor.c, relax.c, p\_lo.c

Performance Contexts:
 

- Citrim 2000 - 16 R110K processors - 800x800
- Power Challenge - 8 R10K Processors - 125x125
- NoW - 8 Sun UltraSparcs - 800x800
- NoW - 4 Sun UltraSparcs - 125x125
- Example: no instrumentation

Routines in Source File: main, MPI\_Comm\_size, MPI\_Comm\_rank, MPI\_Get\_processor\_name, fprintf

Routines in Performance Data:
 

- MPI\_Sendrecv
- MPI\_Send
- MPI\_Reduce
- MPI\_Bcast
- MPI\_Recv

Source File: /home/freed/derose/DemoPablo/Install/SourceFiles/CMPI\_RBSOR/prbsor.c

```

/*
 * The following line has statements grouped together
 * to test some functionalities of the svPablo GUI.
 */
for (i=1; i<it; i++) { relax(n,omega,f,u,smynorm); update
MPI_Sendrecv( &u[myend*n], n, MPI_DOUBLE, botton, (myid+1
u, n, MPI_DOUBLE, top, myid * blocksize,
MPI_COMM_WORLD, &status );
MPI_Sendrecv( &u[n], n, MPI_DOUBLE, top, myid * blocksize
&u[(myend+1)*n], n, MPI_DOUBLE, botton,
(myid+1) * blocksize + 1, MPI_COMM_WORLD, &
if (myid == 0) oldNorm = norm;
MPI_Reduce(&smynorm, &norm, 1, MPI_DOUBLE, MPI_SUM, 0, MPI
saveOutput( u, n, i, norm, oldNorm );
MPI_Finalize();

```

Performance metrics

Color boxes are clickable

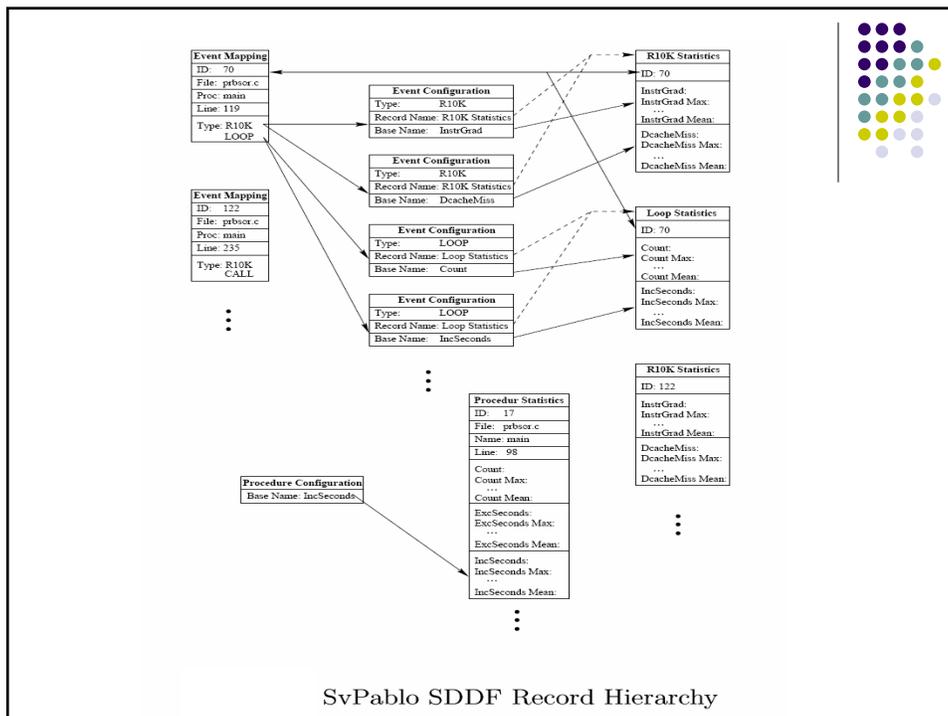
## Details: How did they achieve that?

- Performance Instrumentation
- Hardware performance integration
- Performance Visualization
- Language and Architecture Transparency

# Language and Architecture Transparency



- Requires a **flexible specification mechanism** for instrumentation points and performance metrics.
- **SDDF (self-describing data format)** separates performance data presentation from language and architecture
  - A group of record descriptors and record instances.
  - Three groups of record descriptors: mapping, configuration, and statistics.



SvPablo SDDF Record Hierarchy

## Language and Architecture Transparency



- SDDF main features.
  - **Portability**: portable directly across systems, in binary or ASCII format
  - **Generality**: a variety of events or data types can be supported
  - **Extensibility**: tool developers can easily add new metrics

## Language and Architecture Transparency



- Event Statistics
  - Performance data is represented by a Event Mapping record and a set of Statistics records
- Procedure Statistics
  - Procedure statistics records define the **performance metrics associated with all procedures**: # of calls to the procedure and the exclusive duration of the procedure. etc.

## Related Works



- Paradyn (U of Wisconsin)
  - a tool for measuring the performance of **large scale parallel programs**
  - Instrumentation and visualization are performed during **run-time**.
- Pablo (U of Illinois)
  - Consists of **several components** for instrumenting and **tracing** parallel programs and for analyzing the trace file.
- AIMS (NASA Ames Research)
  - A tool kit for parallel applications.
  - Support **MPI and PVM**.

## SvPablo Summary.



- **A graphical environment** for instrumenting application source code and browsing dynamic performance data.
- Support for not only software performance measurement but also access to **hardware performance** counters.
- **Language and architecture transparency** by representing performance data via XML-like format.