Performance Measurement and Prediction

Luiz DeRose
Dan Reed
University of Illinois
Presentation Outline

- Source code performance prediction
  high-level language integration
  symbolic performance model
  performance prediction experiments

- SvPablo performance browser
High-Level Language Integration

**Motivations**
- emerging high-level languages (HPF and HPC++)
- aggressive code transformations for parallelism
- large semantic gap between user and code

**Goals**
- relate dynamic performance data to source
- generate instrumented executable/simulated code
- support performance prediction
- support scalability analysis
High-Level Language Integration

HLL Source Code

Performance Specification

Other Compilers

Polaris

HPC++ Preprocessor

Instrumented CORBA ORB

Instrumented Java VM/RMI

Instrumented Executable

Symbolic Models

Nexus/Globus Substrate

“Scalable” SMP

Paradyn

Pablo Autopilot

Virtue
Symbolic Performance Prediction

- **Rationale**
  - rapid assessment of design alternatives
  - identification of performance bottlenecks

- **Approach**
  - compile-time generation of cost expressions
  - augmentation with selected measurements
Fortran 95D Symbolic Prediction

- Frontend
- Parameter Extraction
- Cost Model Construction
- Raw Cost Model
- Symbolic Cost Model
- Communication Placement
- Communication Refinement
- Computation Partitioning
- Code Generation
Performance Prediction Model

- Computation
- Memory
- Communication
- I/O

Predicted performance
Symbolic Performance Prediction

- Two phase prediction approach
  - static traversal of compile-time AST
  - dynamic integration of measured data

- AST traversal
  - accumulate operation and memory access counts
  - aggregate for block statements
  - loops and conditional statements
Symbolic Performance Prediction

- Dynamic data integration
  - loop branch estimates
  - measured operation costs
- Performance predictions
  - instantiate array sizes (loop bounds)
  - specify number of processors
  - evaluate symbolic expression
SvPablo/Polaris Performance Prediction

Static Phase

- Source Program
- Instrumented Source
- Native Compiler
- Polaris Analysis
- Symbolic Expressions
- Executable Program
- Capture Library
- Program Execution
- Data Set
- Performance Data
- Merged Model
- Hardware Costs
- Program
- SvPablo
Hardware Cost Variables

- Addition and Subtraction by type
- Logical Operation
- Multiplication by type
- Division by type
- Intrinsic Functions by type and class
  (e.g., Min and Max, Trigonometric Functions, etc.)
- Load by number of bytes and rank
- Store by number of bytes and rank
- Overhead (Loop and Conditional Statement)
Symbolic Cost Expression

Loop 420 (HYDRO-2D)

DO 420 j=1,N
  DO 420 i=1,M
    C = dmin1(RP(i,j+1), RM(i,j))
    IF ( AR2(I,j) .LT. 0.0D0 ) THEN
      C = dmin1(RP(i,j), RM(i,j+1))
    ENDIF
    AR2(i,j) = C * AR2(i,j)
  420 CONTINUE

Cost =
(N+1) * Loop Overhead +
M*N * (1+%True) * Int Add +
M*N * Dbl Mul +
M*N *(1+%True)* Intr Dbl Min+
(N+1) * (1+M*(4+2*%True) * Load Scalar Int +
M*N * Load Scalar Dbl +
M*N *(3+2*%True)*Load 2D Dbl +
(N+1) * Store Scalar Int +
M*N *(1+%True)*Load Scalar Dbl+
M*N * Store 2D Dbl
SWIM Performance Predictions
TOMCATV Predictions

![Bar chart](chart.png)

- **Execution Time (sec)**
- **Measured Execution**
- **Predicted Memory**
- **Predicted Computation**

- **MAIN_do100**
- **MAIN_do120**
- **MAIN_do130**
- **MAIN_do60**
- **MAIN_do80**
Instrumentation Integration

- Leverage best toolkit features
  - Paradyn
    - dynamic object code patching
    - standard software metrics
    - hardware performance data support
  - Pablo/Autopilot/Virtue
    - real-time data analysis
    - flexible data metaformat
    - adaptive resource control
    - performance visualization
SvPablo Performance Browser

- **Instrumentation**
  - interactive
    - ANSI C
    - Fortran 77/Fortran 90
  - automatic
    - PGI HPF

- **Data capture**
  - dynamic software statistics
  - SGI R10000 counter values
SvPablo Code Browser

“Instrumentable” Constructs

```c
MPI_Init( argc, argv );
MPI_Comm_size( MPI_COMM_WORLD, &numprocs );
MPI_Comm_rank( MPI_COMM_WORLD, &myid );
MPI_Get_processor_name( processor_name, processor_name );
#endif
#endif
#endif
#endif
#endif
#endif
#endif
#endif
#endif
#endif
#endif
#endif
```

Project Description: Red Black SOR in C using MPI

Performance Contexts:

- Origin 2000 - 16 R10K processors - 800x800
- Power Challenge - 8 R10K Processors - 125x125
- NoW - 8 Sun UltraSparcs - 800x800
- NoW - 4 Sun UltraSparcs - 125x125

Sample code instrumentation
SvPablo Code Browser

Metrics
SvPablo Code Browser
SvPablo Code Browser
SvPablo Language Transparency

- Meta-metaformat for performance data
  - language defined by line and byte offsets
  - metrics defined by mapping to offsets
  - SDDF records
    - performance mapping information
    - performance measurements

- Result
  - language independent performance browser
  - mechanism for scalability model integration
SvPablo Language Transparency

- Three SDDF record types
  - configuration
    - definition of performance metric set
    - set may contain multiple metrics
  - mapping
    - source code location
    - “pointers” to actual set of performance statistics
  - statistic
    - one for each performance metric
    - contain actual performance data
SvPablo SDDF Record Hierarchy

Event Mapping
ID: 70
File: prbsor.c
Proc: main
Line: 119
Type: R10K LOOP

Event Configuration
Type: R10K
Record Name: R10K Statistics
Base Name: InstrGrad

Event Configuration
Type: LOOP
Record Name: Loop Statistics
Base Name: Count

Event Mapping
ID: 122
File: prbsor.c
Proc: main
Line: 235
Type: R10K CALL

Event Configuration
Type: R10K
Record Name: R10K Statistics
Base Name: DcacheMiss

Event Configuration
Type: LOOP
Record Name: Loop Statistics
Base Name: IncSeconds

R10K Statistics
ID: 70
InstrGrad: ...
InstrGrad Max: ...
...
InstrGrad Mean: ...
DcacheMiss: ...
DcacheMiss Max: ...
...
DcacheMiss Mean: ...

Loop Statistics
ID: 70
Count: ...
Count Max: ...
...
Count Mean: ...
IncSeconds: ...
IncSeconds Max: ...
...
IncSeconds Mean: ...
SvPablo Directions

- **Infrastructure extensions**
  - explore port to Windows NT
  - add Pentium counter support (Paradyn)

- **Scalability extensions**
  - extend symbolic prediction model
  - integrate predictions using SvPablo transparency