Paradyn Parallel Performance Measurement Tool

Barton P. Miller, et al.
University of Wisconsin, Madison

Presented by: Hamid

Motivation

- To measure the performance of long running programs on large machines using large datasets

The Challenges

- Scalability:
  - Large programs
  - 100's or 1000's of nodes
  - Long runs (hours or days)

- Automate Tuning Process:
  - Simplify the task of programmer
  - Deal with increasing complexity

- Support Heterogeneity:
  - Clusters, SMFs
  - OS: UNIX, NT, Linux
  - Various languages

- Extensible (Open Interfaces):
  - Incorporate new sources of performance data
  - Include new visualizations

Features

- Data Abstractions
  - Metric-focus grids
  - Time histograms

- Support High-level Parallel Languages
  - Relating low-level problems to high-level source code

- Streamlined Use
  - Easy installation
  - Avoiding recompile and relink

Searching for Bottlenecks

1. Start with coarse-grain view of whole program performance
2. When you see a problem, collect more information to refine this problem.
3. Repeat step #2 until you have a precise enough cause.
4. Collect information to try to refine to particular hosts, processes, modules, functions, files, etc.
5. Repeat step #4 until you have a precise enough location.

This type of iteration can take a user many runs of a program to reach a useful conclusion.

Approach the Problem Differently: Do Everything Dynamically

Paradyn allows the programmer to do this on-the-fly in a single execution.
Paradyn Overview
Barton P. Miller

The Major Technologies

- Dynamic Instrumentation
  - On-the-fly: Insert, remove, and change instrumentation in the application program while it is running.

Automating the Search for Bottlenecks
- The Performance Consultant: identify bottlenecks and automatically control the Dynamic Instrumentation.

Basic Abstractions
- Metric-focus grids (matrix)
  - Metrics: CPU time, memory usage, message rate, IO rate, floating point operations
  - Focus: procedures, processor nodes, threads, disks, barriers
- Time histograms
- Grid values: time histograms or single values

Paradyn Architecture

Paradyn Configuration File (PCL)
- Describes:
  - Architecture
  - Operating system
  - Environment characteristics
  - New metrics
  - New visualizations

Dynamic Instrumentation
- Does not require recompiling or relinking
  - Saves time: compile and link times are significant in real systems.
  - Can profile without the source code (e.g., proprietary libraries).
  - Can profile without linking (relinking is not always possible).
- Instrument optimized code.
Dynamic Instrumentation (con'd)
- Only instrument what you need, when you need
  - No hidden cost of latent instrumentation.
  - Enables "one pass" performance sessions.
- Can monitor running programs (such as database servers)
  - Production systems.
  - Embedded systems.
  - Systems with complex start-up procedures.

Dynamic Instrumentation (con'd)
- Anything in the application's address space can become a performance measure
  - Application metrics: transactions/second, commits/second.
  - OS metrics: page faults, context switches, disk I/O's.
  - Hardware metrics: cycle & instruction counters, miss rates, network statistics.

Basic Instrumentation Operations
- Points: places to insert instrumentation
- Primitives: code that gets inserted
  - Counters
    - Set counter
    - Add to counter
    - Subtract from counter
  - Timers
    - Set timer
    - Start timer
    - Stop timer
- Predicates: condition of primitive execution (if)

Basic Instrumentation Operations
- Combining primitives & predicate to create metrics
- Time spent sending messages by procedure foo and its descendants:

Instrumentation Generation
- PCL files describe the translation of metric-focus specifications to points, primitives and predicates
- Instrumentation manager performs this translation
- Points are:
  - Procedure entries
  - Procedure exits
  - Procedure call sites

Decision Support ("Performance Consultant")
- W3 search model, answers the questions:
  - Why is the program running slowly?
  - Where in the program is this occurring?
  - When does this problem occur?
- We create a regular structure for the causes of bottlenecks.
  This makes it possible to automate the search for bottlenecks.
The "Why" Axis: A Hierarchy of Bottlenecks
- Potential bottlenecks are represented as hypotheses.
- Evaluating hypotheses triggers dynamic instrumentation.
- Bottlenecks are based on user-set thresholds: Total sync blocking time < 25% of exec time

"Why" Axis: A Hierarchy of Bottlenecks
- TopLevelHypothesis
  - Sync
  - CPU
  - I/O
  - Memory

"Where" Axis: Hierarchy of Classes

Call Graph

Call Graph Based Performance Consultant Example

Top Level Hypothesis
- SyncWaitBound
- I/OWaitBound
- CPUBound
Call Graph Based PC Example

Top Level Hypothesis

- CPUBound
- SyncWaitBound
- I/OWaitBound
- main

Call Graph Based PC Example

Top Level Hypothesis

- CPUBound
- SyncWaitBound
- I/OWaitBound
- main

Call Graph Based PC Example

Top Level Hypothesis

- CPUBound
- SyncWaitBound
- I/OWaitBound
- main

Call Graph Based PC Example

Top Level Hypothesis

- CPUBound
- SyncWaitBound
- I/OWaitBound
- main

Performance Consultant search begins...

- Hypothesis colors: Gray, green, blue, pink
- Arc colors: Why, where, when axis
Performance Consultant

Performance Consultant module of Paradyn automatically searches the W3 search model and refines the search.

Performance Consultant 1st refinement...

Performance Consultant 2nd refinements...

Performance Consultant search complete

Performance Consultant complete (compact view)

Metric/Focus and Visualization Selection
Paradyn Overview
Barton P. Miller

8 November 2005

Paradyn assorted time-history & summary visis

Bar Chart Display

DynInst API: A Common Substrate

New Runtime Tool

Paradyn and DynInst on the AIX/SP

Both Paradyn and DynInst fully operational on AIX and SP.

IBM is developing new products based on this technology under the name DPCL.

How to Get a Copy of Paradyn:

Release 4.2:
- Free for research use.
- Also runs on Solaris (SPARC & x86), NT (x86), Irix, Linux (x86), DEC Unix.

http://www.paradyn.org/
paradyn@cs.wisc.edu
Questions?