Performance analysis support on the Operating Environment

Marisa Gil, Xavier Martorell (marisa, xavim)@ac.upc.es
Curs de doctorat 2002/2003
Departament d’Arquitectura de Computadors
UPC

Motivation

- Complexity grows in all the areas
  - Processors
  - Computers
  - Operating System
  - Run-time libraries
  - Applications

- We have a problem...
Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions

Motivation

Why such a loop is performing so bad???

!$OMP PARALLEL DO
do i = 1, N
  do j = 1, N
    A(i,j) = 0.0

- Is it a matter of TLB misses?
- Primary data cache misses?
- External interventions???
- False sharing?
**Motivation**

- Can we obtain statistics about performance?
  - Resource utilization
    - Caches
    - Memory
    - FP instructions

**Introduction**

- Tracing is necessary to figure out about performance
- Need for low interference
- Two views
  - Internal to the application
  - External from the operating system
    - Several applications executed at a time
Architectural support

- **Performance event counters**
  - User
  - Kernel
  - Exception

- **Limitations**
  - Small number of registers to count on
    - Multiplex
  - Small register size (R10K)
    - Overhead
  - Hardware dependent
    - Processors from the same family have different counters

OS support

- **None (...)**
  - Currently, Linux

- **Reading/writing global counters**
  - Some support libraries

- **Performance counters on thread/process context**
  - IRIX
OS support

- Process/thread information
  - /proc
  - system calls

- Memory/swap usage

- [Processor information]
  - [Which process is it executing?]
  - How many time is spent in OS/exception code

OS support

- Availability of information related to processors
  - Time spent in... [sysmp (MP_SAGET, SINFO_CPU, &data)]
    - User level
    - System level
    - Waiting (IO, SWAP, PhysIO, Graphics...) [SGI]
    - Memory management
    - Interrupt
    - Idle
  - [Process currently executed]
    - Not usually available :-(

*Dept. d'Arquitectura de Computadors*
### RTL support

#### Counting
- User-defined event-sets
  - Groups of counters got as a unit of information
  - Move counters as a bulk across the OS interface
  - Specialize use of counters and relate different ones
- Multiplex
- Overflow detection
  - User-programmed routines

#### Multiplex
- Limited number of events counted at a time
- Software can switch events counted
  - Resolution depends on interval timers
- Overhead in the implementation
- Inaccuracy
  - Short high performance routines hidden
**RTL support**

- **User-defined handlers**
  - Activated on an event exceeding a specific threshold
  - Usually through OS support
    - Otherwise an alarm handler can help
  - Implemented through signals... efficient???

- **Getting/extracting the information**
  - From the kernel
    - Specific system call needed
    - Reading /proc
  - Directly from the processor
    - Such machine language instructions are usually privileged
    - Pentium architecture allows them to be read from user-level
      - Setting a bit in the processor (CR0?)
    - Always will read own counters
Support to the end-users

- Profiling tools
  - Traces are analyzed/visualized by end-user utilities
  - Prof generates histograms
    - Number of events per subroutine/program line
  - PARAVER shows the exact behaviour of the application
  - Vampir

Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions
**Goal**
- Specify a standard API for accessing hardware performance counters

**Defines**
- Standard set of hardware events
- Library interface

**Architecture**
- PAPI low level
  - Multiplex counters
  - Overflow management
  - Timer interrupt
- PAPI machine dependent substrate
- Kernel extension
- Operating System
- Performance Counter Hardware
- PAPI high level
How to use

Performance Analysis Tools
Feedback-Directed Compilers
Adaptive Run-Time Libraries
Application measurement and timing

PAPI low level
Multiplex counters
Overflow management
Timer interrupt

PAPI high level

Processor / OS supported

- POWER3, AIX4.3
- POWER4, AIX5
- CRAY T3E, Unicos
- AMD Athlon, Pentium III, Linux [patched]
- Pentium III, Windows
- Itanium I & II, Linux [patched]
- UltraSparc I, II & III, Solaris 2.8
- MIPS R10K, R12K, IRIX
- Alpha, Tru64 UNIX
- Alpha, Linux [patched?]
Overview of the interface

- Set of hardware independent events
  - Not available in all hardware
  - Check list!
- Low level API
  - Counter management and acquisition
- High level API
  - Easier counter acquisition

Low-level PAPI

- `PAPI_create_eventset (eventSet)`
  - Creates an eventSet. Not thread-safe
- `PAPI_destroy_event_set (eventSet)`

- `PAPI_add_event (eventSet, event)`
- `PAPI_add_events (eventSet, events, number)`
- `PAPI_cleanup_eventset (eventSet)`
- `PAPI_rem_event (eventset, event)`
- `PAPI_rem_events (eventSet, events, number)`
  - EventSet management
Low-level PAPI

PAPI_get_hardware_info (hw_info)

PAPI_get_executable_info (exe_info)

PAPI_get_opt (option, value)
  - Access to the environment

PAPI_query_event (eventCode)
  - Checks whether the hardware support such event

PAPI_set_domain (domain)
  - Execution domain to count
    - User, kernel...

PAPI_set_granularity (granularity)
  - thread/process/process group/current cpu/all cpus

PAPI_get_overflow_address (context)
  - Gets overflow PC

PAPI_overflow (eventSet, eventCode, threshold, flags, handler)
  - Sets the overflow handler for the indicated event in eventSet

PAPI_set_multiplex (eventSet)
  - Allows multiplexion

PAPI_get_real_cyc (long long)

PAPI_get_real_usec (long long)
  - Time control
Low-level PAPI

- **PAPI_start (eventSet)**
  - Starts counting...not if in conflict with another eventSet

- **PAPI_stop (eventSet, values)**
  - Stops the eventSet and reads values

- **PAPI_accum (eventSet, values)**
  - Accumulates current counters in eventSet to values

- **PAPI_read (eventSet, values)**
  - Copies current counters in eventSet to values

- **PAPI_write (eventSet, values)**
  - Copies the values to the current counters in eventSet

High-level PAPI

- **PAPI_start_counters (events, number)**
- **PAPI_read_counters (values, number)**
- **PAPI_accum_counters (values, number)**
- **PAPI_stop_counters (values, number)**
- **PAPI_flops (...)**
Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - IntoneOMPI tracing
  - Smpus
- Conclusions

Rabbit

- Thesis
  - “All programs should be designed to measure their own performance”
- Precondition
  - “Performance measurement software should be completely portable”
- Preclusions
  - “Performance measurement hardware and operating system interfaces are completely non-portable”
Rabbit

- **Supports Pentium/AMD**
  - Different hardware
    - Pentium has two 40-bit counters
    - AMD has two 48-bit counters
  - Different event sets
  - Single register counting cycles (64 bits)

- **Supports Linux**
  - No patch
  - On dual/quad -processor systems
    - counters are lost on process migrations

- **Simple interface**
  - `pmc_open/pmc_close`
  - `pmc_start`
  - `pmc_select`
  - `pmc_read`
  - `pmc_counter_init`
  - `pmc_accumulate`
Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions

libFASTparparaver

- Merges
  - tracing applications
  - hardware event counters
- Visualization in Paraver
libFASTparparaver

- **Interface**
  - Compiler oriented
    - pushstate(newstate)
    - popstate()
    - changestate(newstate)
    - changeandevent(newstate, event, value)
    - pushandevent(newstate, event, value)
    - popandevent(event, value)
    - userevent(event, value)
    - eventcount(userevent)
    - endeventcount(userevent)
  - Internal (user oriented?)

- **Counters sampled implicitly**
  - Using IRIX/AIX system interfaces

libFASTparparaver

- **What can we get?**
  - thread status
  - Counter values
  - Source code line
Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions

Intone OMPI

- Application/counting events related to source code
  - Automatically
- OMPI interface
  - OpenMP constructs
  - Functions
  - Event counters
  - Handles and tags
  - Source code locations
**OpenMP constructs**

- `ompi_enter_parallel (parHandle, regionTag, numThreads, sclHandle)`
- `ompi_leave_handle (sclHandle)`
- `ompi_enter_openmp (eventHandle, tag, sclHandle)`
- `ompi_leave_openmp (sclHandle)`
  - DO, SECTION, BARRIER, SINGLE, MASTER, CRITICAL...

- `ompi_define_parallel (regionName, parHandle)`

**Functions**

- `ompi_enter_function (fHandle, sclHandle)`
- `ompi_leave_function (sclHandle)`
- `ompi_define_function (functionName, classHandle, fHandle)`
**Event counters**
- `ompi_log_counter (noCounters, counterHandles, values, sclHandle)`
- `ompi_define_counter (counterName, type, unit, bounds, counterHandle)`
- `ompi_set_counter_callback (fCallback, custom, nCounters)`

**Handles and tags**
- `ompi_define_tag (tagString, tagHandle)`

**Source code locations**
- `ompi_define_scl (fileName, lineNumber, sclHandle)`
Intone OMPI

Visualization

Counting iterations (or “find the 8 differences” - I)
Counting iterations (or “find the 8 differences” - II)

Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTpararaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions
Scpus

- Allows tracing of OS view
  - Externally to applications

- Searches /proc for any useful information
  - Mapping of processes/threads to processors

- Gets other information from the system
  - Processor states

- Would be useful to have the process running in each processor as a OS primitive!!!

Scpus

- Visualization

![Visualization Diagram]
Scpus

- Poor coordination
- High number of movements
- Higher execution time in kernel mode
- Worse execution times and throughput

- Better coordination
- Better processor to cpu mapping
- Consistent execution times
- Improved memory behaviour
  - Local accesses
  - Lower kernel mode execution time

Dept. d’Arquitectura de Computadors

Entornes Operatius per la Gestió de Recursos...
Outline

- Motivation
- Introduction
- Support to performance analysis
  - Architectural, OS, RTL
- Examples
  - PAPI
  - Rabbit
  - libFASTparparaver / Ompitrace
  - Intone OMPI tracing
  - Scpus
- Conclusions

Conclusions

- Complexity grows in all areas
  - Processors
  - Computers
  - Operating System
  - Run-time libraries
  - Applications

- Have we solved that problem???
  - Or at least, do we have now some clues on how to deal with it?
Conclusions

- **Multiple tools support performance analysis**
  - Libraries
    - PAPI
    - Rabbit
    - libFASTparparaver
  - End-user tools
    - Visualization

- **Internal and external views**
  - Application oriented
  - Workload oriented

Support libraries

- **List...**
  - Papi http://icl.cs.utk.edu/projects/papi
    - Innovative Computing Lab (Computer Science, University of Tennessee)
      - Jack Dongarra,
  - Rabbit http://www.scl.ameslab.gov/Projects/Rabbit
    - Scalable Computing Lab (Ames Lab, US DOE, Iowa State University)
      - Don Heller
Support libraries

**List...**
- Vtune
  [link](http://developer.intel.com/software/products/vtune/index.htm)
  ✔ Intel
- DCPI [link](http://www.tru64unix.compaq.com/dcpi)
  ✔ Compaq (Digital)

**Other**
- Parallel Tools ([link](www.ptools.org))
- Solaris hardware statistics tool