Perfmon2, PMU of CellBE & PPC970MP PMU and Perfminer

Philip Mucci

Barcelona Supercomputing Center

2/11/06

mucci at cs.utk.edu

Linux Kernel Support for PMU's

- None, up until now.
- Historically, only IA64 and whatever was necessary to support Oprofile.
  - Oprofile is a root level tool, only for measuring counter overflow.
  - Great for system tuning, not for in-production systems.

Linux Kernel support for PMU's

- This is rapidly changing. IA64 support has evolved into a generalized infrastructure for PMU support called Perfmon2.
- Under active development for 2-3 years now, with a long history prior based on IA64.
- Work done by Stefane Eranian of HP (and one of the two Linux/IA64 kernel architects.)

Linux Kernel support for PMU's

- Perfmon2 undergoing active development by Stefane, with some help from myself and others.
- Actively being reviewed by LKML and piece by piece, is being accepted into the mainline.
- Current support: x86, x86_64, MIPS and IA64.

Perfmon2

- Measurement types
  - Counting
  - Sampling
- Scopes
  - System-wide
  - Per-thread
- Views
  - First person
  - Third Person
- Integration
  - Cooperates with Oprofile

The following slides borrow heavily from Stefane Eranian's talk at OLS2006 at http://perfmon2.sourceforge.net/ols2006-perfmon2.pdf

Perfmon2 (2)

- Counters virtualized to 64-bit
- Logical view of PMD's and PMC's, not machine specific.
- System call approach rather than driver approach.
- Compatible with existing mechanisms
  - – Mmap, signals, ptrace, etc...

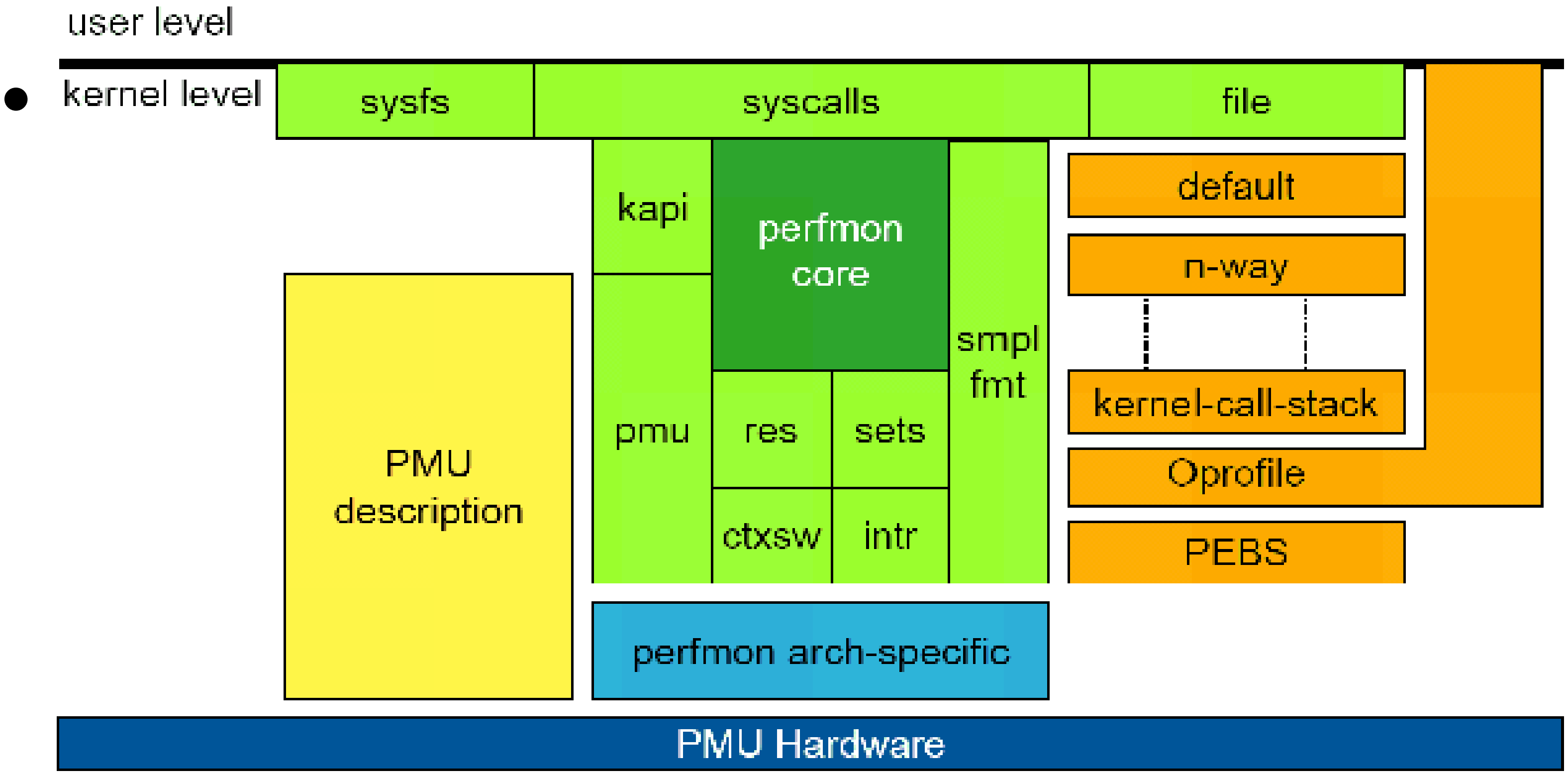
Perfmon2 Sampling

- Traditionally sampling has been looking at the IP upon PMD interrupt, passed through to the user through a signal context.
- IA64 and PPC64 series introduced address and branch sampling.
- Perfmon2 provides access to buffered, customized sampling of any PMU resource.

Perfmon2 Multiplexing

- PAPI has had the ability to multiplex counters for a while, but it does this at user level with signals and a timer.
- Perfmon2 can do this in the kernel.
  - Much lower overhead.
  - Less pollution of user counts.
  - Provides switching based on PMD overflow or clock.

Perfmon 2 Architecture Summary



Blatantly stolen with permission from Stefane Eranian's
talk at OLS2006 at http://perfmon2.sourceforge.net/ols2006-perfmon2.pdf

Perfmon2 and PPC64/CellBE

- Preliminary work done 6 months ago on Perfmon2 for the Power 5.
  - Non-working, non-maintained.
- No work as of yet done on CellBE.
  - Perhaps this/next week a framework.
- Issue is primarily a lack of dedicated root access to systems.

Perfmon2 and IBM

- IBM has (in correspondence) recognized that Perfmon2 is the future.
- They are looking forward to a PPC64 and CellBE port.
- Unknown status, financial commitments, resource allocations.

PPC970 and Perfmon2

- Why is this important?
  - Because there are many features of the PPC970 series that are useful.
  - Future upgrades of MareNostrum may not have the adequate support in PerfCtr (or the resources necessary to do so).

The 970MP PMU

- 8 performance counters + SIAR and SDEAR
- Sampling support
- Event filtering
- Instruction matching
- Thresholding
- Triggering

SIAR and SDAR

- Registers that contain the exact IP and effective address of the instruction.
- For example, IP of load, address of load for a L2 cache miss.
- Additional qualification adds the ability to sample based on threshold. (Misses > N cycles)

PPC970 stall events

- Provides a full complement of stall events
  - LSU, FXU, D-Cache, FPU, FXU long, reject, I-Cache and others.
  - Can be translated to cycles with edge detection features.
  - Stall cycle accounting is becoming more important than cycle counting because events can hide.

Perfmon2 and libpfm

- Perfmon2 provides the means to program the registers.
- It does not dictate the register contents!
- This is often even more work than getting the kernel components correct.
- Perfmon2 comes with libpfm to help.

PPC970 and Perfmon2

- We (myself), BSC and IBM should work together to get a working Perfmon2 and Libpfm port to MareNostrum.
- Currently, there is no plan for this, AFAICT.

CellBE PMU

- 8 16-bit counters or 4 32-bit counters
- Event event can count: events, cycles event active or inactive, cycles until (trigger)
- Start/stop qualifiers, thresholds
- Over 400 events can count anything from PPU, PPSS, 8SPU, 8MFC, EIB, MIC, BIC or IOC.

CellBE PMU Trace Array

- 1024 128-bit entries which generates an interrupt to the PPE when full.
- Automatically filled according to a specified interval.
- Transfered to main memory through polled I/O or DMA.

CellBE PMU

- Of most interest are the events for each SPU.
  - Instructions (single/dual/pipeline)
  - Load/store
  - Branch stalls, mispredictions
  - Full complement of stall events

Programming the CellBE PMU

- Connection between the PMU and the various 'islands' is quite complex through the use of the debug bus.
- Care must be taken to avoid programming.
  - Libpfm is going to be a lot of work.
- SPE's have no access to the PMU, the PMU is in the 'pervasive logic'.

Linux and the CellBE PMU

- Current 2.6.16 kernel has some support software for programming the PMU.
  - Used only by oprofile, no system-call API.
- This software count be leveraged to support a version of Perfmon2.

Perfmon2 and CellBE PMU

- From the example code in oprofile, programming the PMU is quite difficult.
- Furthermore, some documentation is missing. “Cell Broadband Engine Book IV”.
- Big challenges:
  - Only 1 PMU for 2 threads, must be shared.
  - Libpfm must handle all dependencies.

PerfMiner and PDC (Center for Parallel Computers)

- The biggest of the centers in Sweden that provides HPC resources to the scientific community. (~1000 procs, ~2TF)
  - Vastly different user bases, from bioinformatics to CCM.
- Wanted to purchase a new machine. (3-4x)
  - Lack of explicit knowledge of the dominant applications and their bottlenecks. No tool!

Related Work Didn't Help

- The same problem over and over: utter lack of detail.
  - Batch logs, SuperMon, CluMon, Ganglia, Nagios, PCP, NWPerf
  - Vendor specific monitoring software...
- Only NCSA's internal system (from Rick Kufrin) met our needs. But not public!

PerfMiner: Bottom Up Performance Monitoring

- Allow performance characterization of all aspects of a technical compute center:
  - Application Performance
  - Workload Characterization
  - System Performance
  - Resource Utilization
- Provide users, managers and administrators with a quick & easy way to track/visualize performance of jobs/system.
- Full transparent integration from batch system to database to web interface.

3 Audiences

- Users: Integrating Performance into the Software Development Life-cycle
  - Quick and elegant way to obtain and maintain standardized perf. information about one's jobs.
- Administrators: Performance Focused System Administration
  - Efficient use of HW, SW and personnel resources.
- Managers: Characterization of True Usage
  - Purchase of a new compute resource.

Site Wide Performance Monitoring

- Integrate complete job monitoring in the batch system itself.
- Track every cluster, group, user, job, node all the way down to individual threads.
- Zero overhead monitoring, no source code modifications.
- Near 100% accuracy.

Batch System Integration

- PDC runs a heavily modified version of the Easy scheduler. (ANL)
  - Reservation system that twiddles /etc/passwd.
  - Multiple points of entry to the compute nodes
  - Kerberos authentication
- Monitoring must catch all forms of usage.
  - MPI, Interactive, Serial, rsh, etc...

Batch System Integration (2)

- Need to a shell script before and after every job.
- We must use /etc/passwd as the entry point!
  - Custom wrapper that runs a prologue and execs the real shell.
  - The prologue sets up data staging area and monitoring infrastructure.
- Batch system runs the epilogue.

Batch System Integration (3)

- Data is dumped into a job specific directory and flagged as BUSY.
- Data about the batch system and job are collected into a METADATA file.

```
JOBID:111714450953
CLUSTER:j-pop
USER:lama
CHARGE:ta.lama
ACCEPTTIME:1100702861
PROCS:4
FINALTIME:1100703103
```

Data Collection with PAPIEX

- PapiEx: a command line tool that collects performance metrics along with PAPI data for each thread and process of an application.
  - No recompilation required.
- Based on PAPI and Monitor libraries.
- Uses library preloading to insert shared libraries before the applications. (via Monitor)
  - Does not work on statically linked or SUID binaries.

Some PapiEx Features

- Automatically detects multi-threaded executables.
- Supports PAPI counter multiplexing; use more counters than available hardware provides.
- Full memory usage information.
- Simple instrumentation API.
  - Called PapiEx Calipers.

Monitor

- Generic Linux library for preloading and catching important events.
  - Process/Thread creation, destruction.
  - fork/exec/dlopen.
  - exit/_exit/Exit/abort/assert.
  - User can easily add any number of wrappers.
- Weak symbols allow transparent implementations of dependent tool libraries.

PapiEx Sample Output

```
PapiEx Version:          0.99rc2
Executable:
/afs/pdc.kth.se/home/m/mucci/summer/a.out
Processor:               Itanium 2
Clockrate:               900.000000
Parent Process ID:       8632
Process ID:              8633
Hostname:                h05n05.pdc.kth.se
Options:                 MEMORY
Start:                   Wed Aug 24 14:34:18 2005
Finish:                  Wed Aug 24 14:34:19 2005
Domain:                  User
Real usecs:              1077497
Real cycles:             969742309
Proc usecs:              970144
Proc cycles:             873129600
PAPI_TOT_CYC:            850136123
PAPI_FP_OPS:             40001767
Mem Size:                4064
Mem Resident:            2000
Mem Shared:              1504
Mem Text:                16
Mem Library:             2992
Mem Heap:                576
Mem Locked:              0
Mem Stack:               32

Event descriptions:
Event: PAPI_TOT_CYC
        Derived: No
        Short Description: Total cycles
        Long Description: Total cycles
        Developer's Notes:
Event: PAPI_FP_OPS
        Derived: No
        Short Description: FP operations
        Long Description: Floating point operations
        Developer's Notes:
```

The Back End

- Directory is marked BUSY with a file.
- After termination of every thread, PapiEX writes a file. (Max 2K in length.)
- At job termination, epilogue script removes BUSY file.
- Data is consumed by an offline process that imports the data to the database and archives the original data on secondary storage.

Scalable Database Design

- Now in version 2, implemented in Postgres.
  - Portable to other back ends.
- May contain many millions of rows for a production system.
- Population by the epilogue is done through Perl scripts and DBI.
  - All DB structure contained in the scripts.
  - No external schemas or DB setup required.

Scalable Database Design (2)

- 4 keys: cluster, job-id, process-id, thread-id
- First version was implemented with a base table of 'standard' metrics and individual tables for specific metrics.
  - Version 2 has a separate table for every metric.
- Each table has a scope (or is a node in an ontology).

Scalable Database Design (3)

- Direct measurements.
  - Events that are measured directly by the underlying performance tool (and METADATA).
- Derived measurements:
  - Events that are explicitly constructed from complex queries.
  - SQL for constructing them is embedded in the database. Measurements can be hidden as VIEWS. Rates, Ratios, etc.

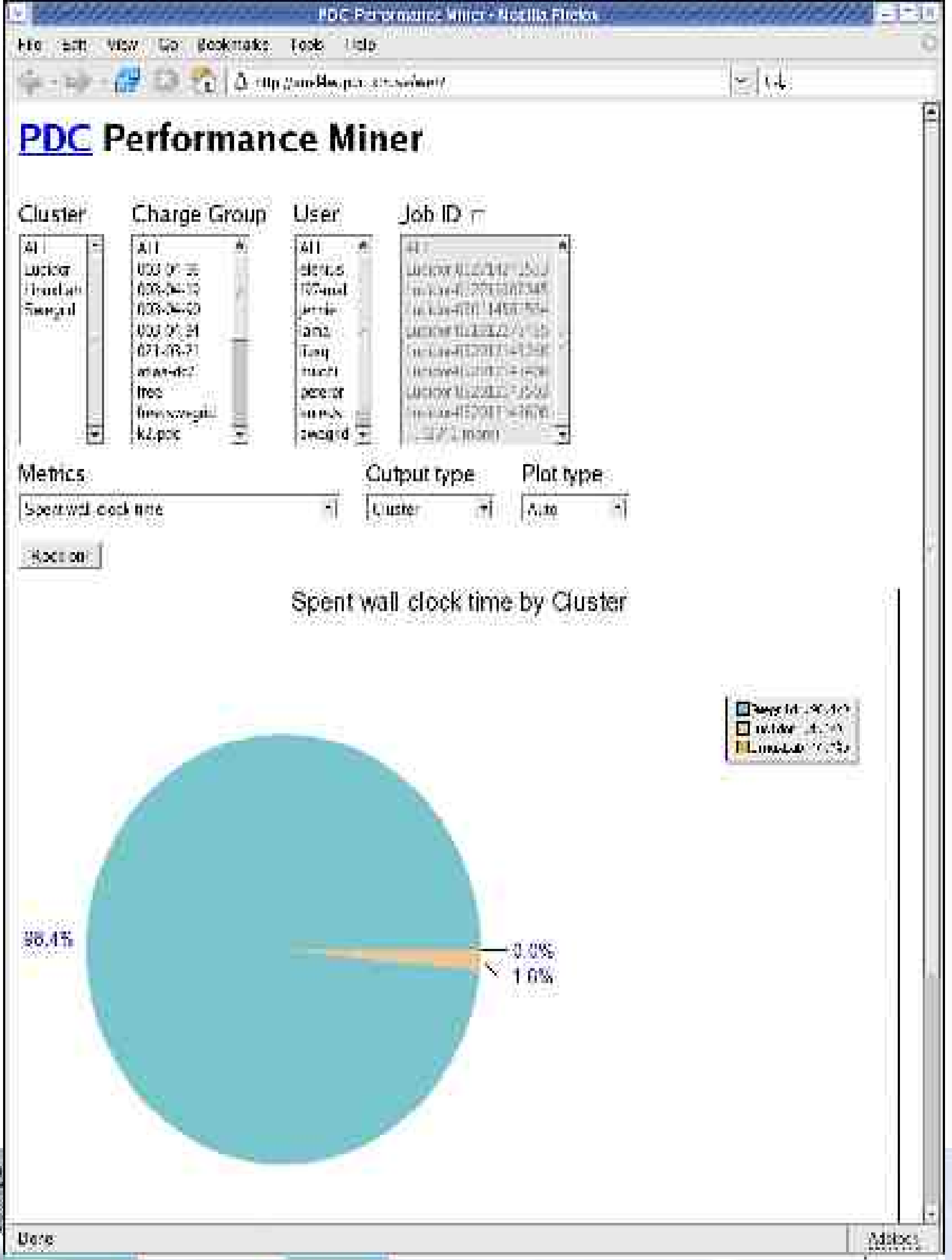
PerfMiner Interface

- Straight HTML fed by PHP scripts.
  - JpGraph/GD and PHP DBI.
- Proof of concept interface: lots more work to do.
- 3 selection criteria:
  - What event to visualize?
  - What range/scope to select?
  - What range/scope to display over?

Test Deployment

- Run for 3 weeks on 3 different clusters.
  - – Lucidor: 90 Dual HP-ZX1 Nodes (IA64)
  - – Swegrid: 100 Pentium 4 Nodes (x86)
  - – Linux Labs: 16 Dual Pentium III (x86)
- ~2.5 Million Threads in the database.

PerfMiner: Main Window



PDC Performance Miner - Mozilla Firefox
File Edit View Go Bookmarks Tools Help

PDC Performance Miner

Cluster: All, Lucidor, [illegible], Swegrid

Charge Group: All, [illegible], free, [illegible], k2.pdc

User: All, [illegible], swegrid

Job ID: All, [illegible]

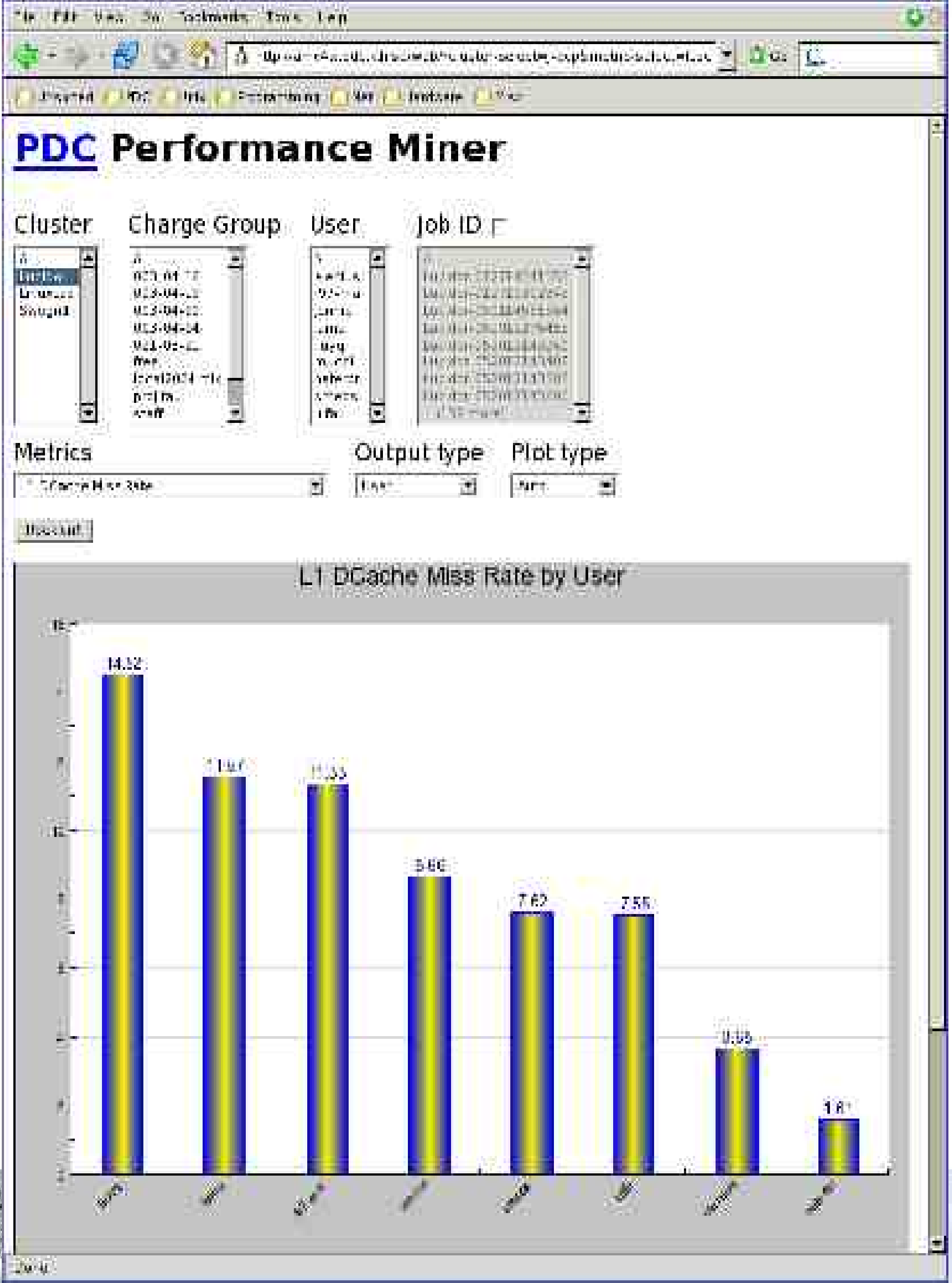
Metrics: Spent wall clock time

Output type: Cluster

Plot type: Auto

Spent wall clock time by Cluster
98.4%
0.6%
1.0%
Done

PerfMiner: L1 Miss Rate by User



PDC Performance Miner

Cluster

Charge Group

User

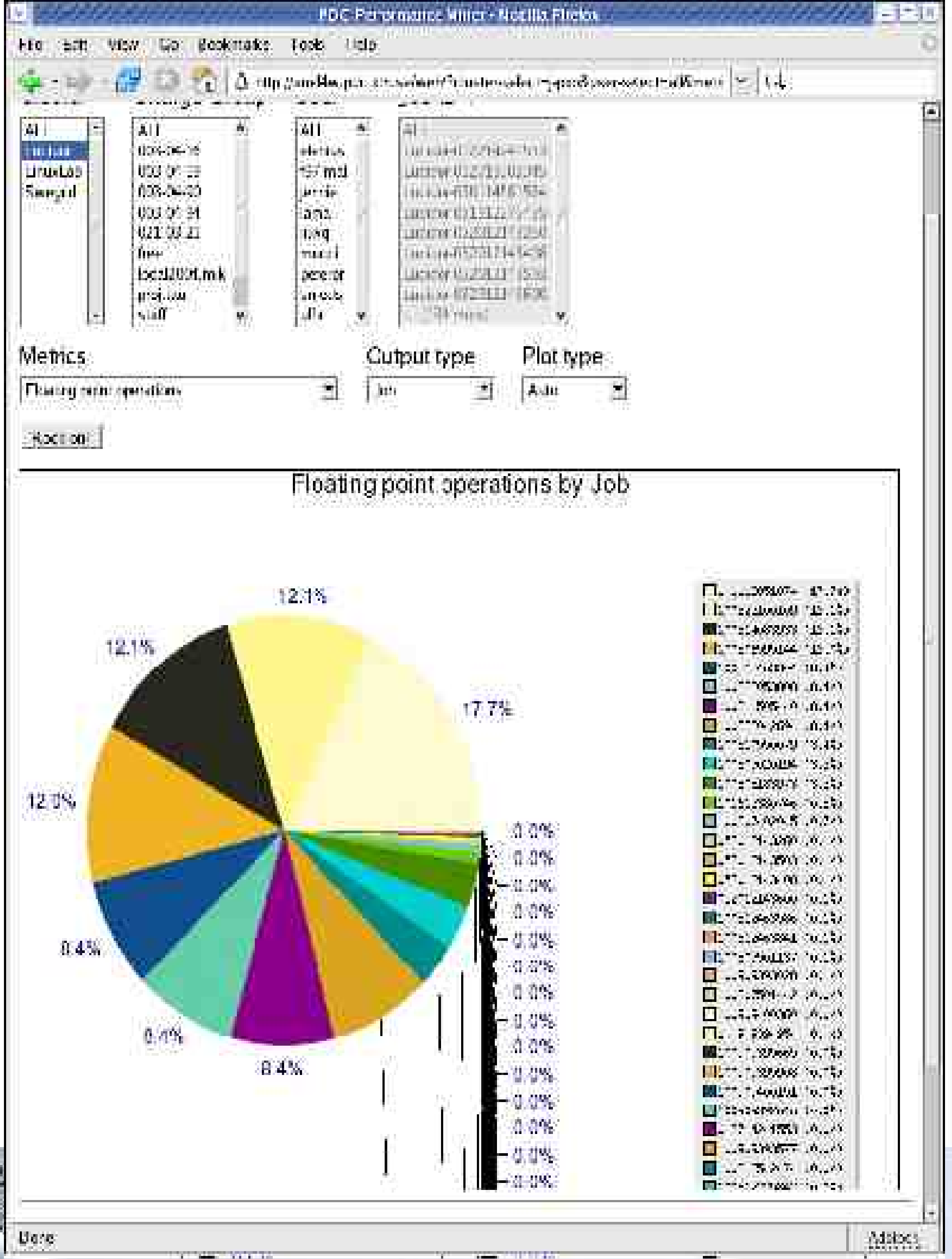
Job ID □

Metrics: L1 DCache Miss Rate

Output type

Plot type

PerfMiner: FP Ops by Job ID

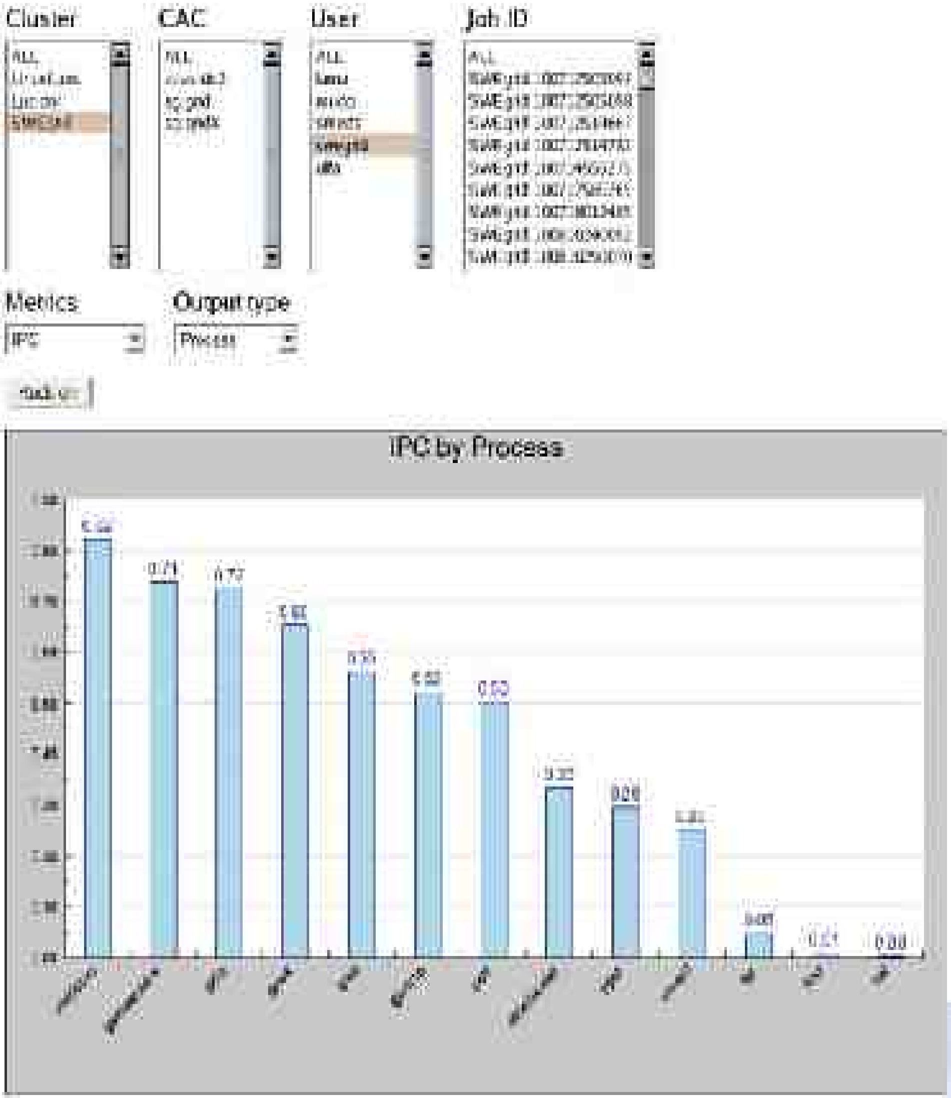


Metrics: Floating point operations

Output type: Job

Plot type: Auto

PerfMiner: IPC by Process

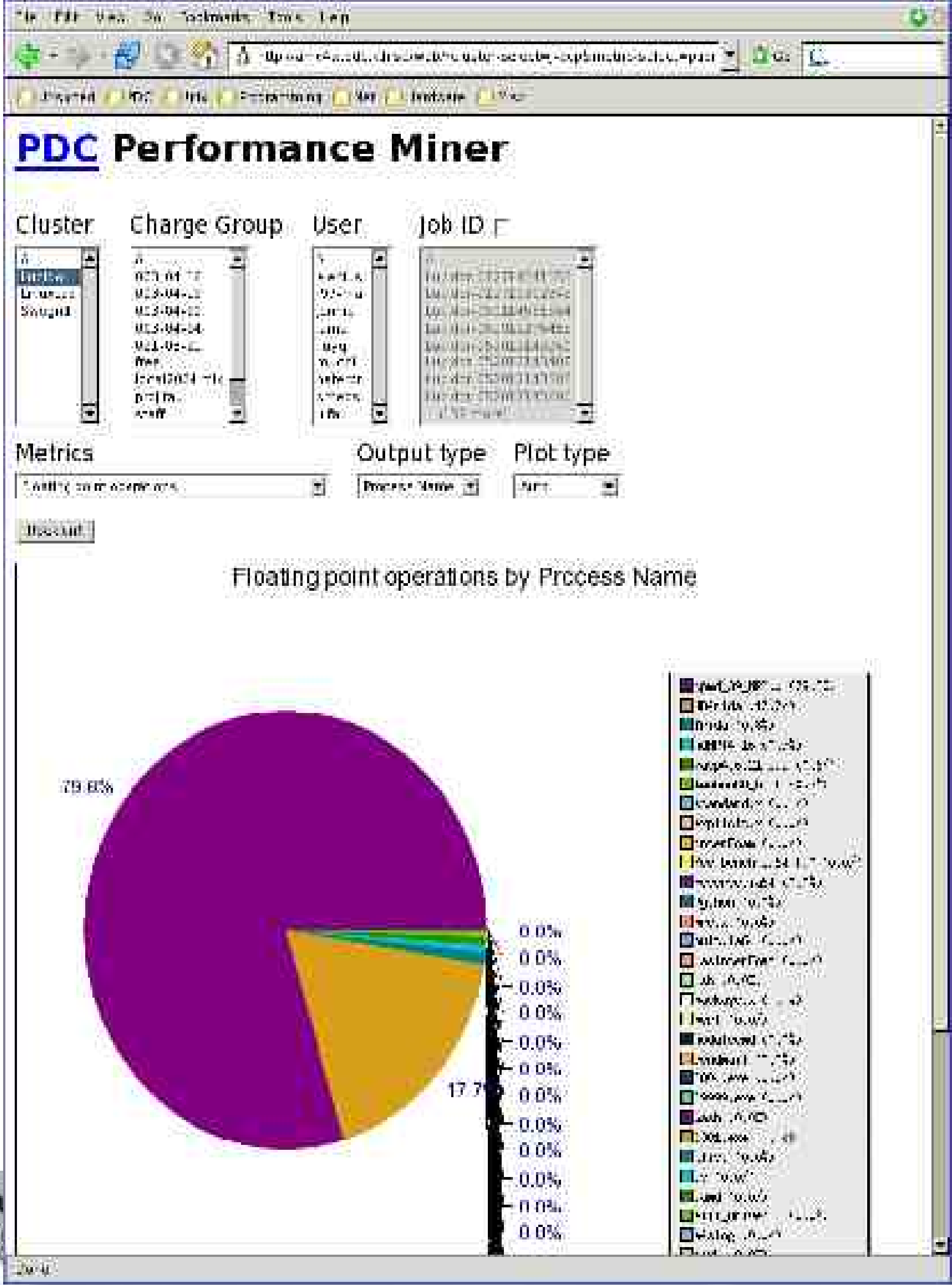


Metrics: IPC

Output type: Process

IPC by Process

PerfMiner: FP Ops by Process Name



Many Additional Queries

- Degree of thread and process parallelism.
  - Wasted nodes, processors?
- Wait time in queue.
- Number of threads in each application.
  - Wasted processor?
- Memory usage.
- Relatively easy to add new complex queries through the derived metrics mechanism.

Issues

- Some queries can take a very long time.
  - – Do the queries need to be realtime?
- Interference with instrumented codes that make use of the PM hardware.
  - – Hints to the batch script to disable monitoring.
- Browser and JpGraph overhead of rendering:

Extensions

- KSOM's, Principle Component Analysis and Data Clustering Techniques.
- Integration with a User and Group Accounting System.
- Additional monitoring modules could be specified by the user.
  - MPI, OpenMP, UPC entry points.
  - User defined entry points.

Portability and Availability

- CVS tree is currently private, but open soon.
  - Contact us if you'd like to be notified.
- System is reasonably portable, but:
  - Datatypes?, VIEW syntax?
  - Batch system and METADATA file are rather specific to the site.
  - You need LD_PRELOAD. (AIX, Unicos)

Scatter Plots and Correlation

- We expect many performance metrics to be more or less correlated in some fashion.
  - (instructions vs. cycles, flops vs. misses)
- Scatter plots provide one method of exposing this.
- Data points that cluster in non-obvious fashion can provide guidance for focused attention.

Links

- http://icl.cs.utk.edu/~mucci/mucci_talks.html
- http://perfmon2.sourceforge.net
- http://icl.cs.utk.edu/~mucci/monitor
- http://icl.cs.utk.edu/~mucci/papiex
- http://icl.cs.utk.edu/papi
- http://perfminer.pdc.kth.se

Questions: mucci at cs.utk.edu & dah at pdc.kth.se