Perfmon2 overview

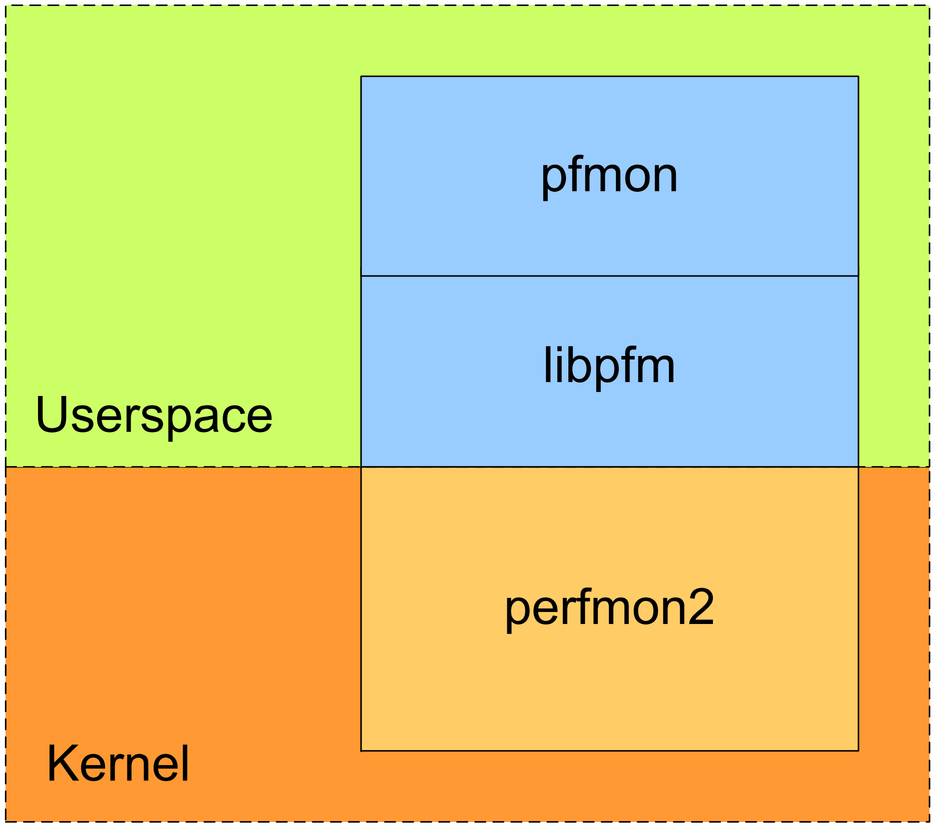
Andrzej Nowak

March 18th 2008

Perfmon2 architecture

Performance monitoring suite for Linux

- **perfmon2 – kernel part**
- **libpfm – userspace interface for perfmon**
- **pfmon – "example" userspace application, perfmon2 client**



Perfmon2

- **Resides in the kernel**
- **Currently available as a kernel patch**
- **Being merged into the Linux kernel mainline**
  - Perfmon2 will eventually be available by default in the Linux kernel (~2.6.26 timeframe)
- **Very basic functionality, keeping the kernel patch slim**
- **Support for numerous architectures:**

  x86, x86-64, ia64, powerpc, cell / ps3, mips, sparc

Pfmon overview

> **Console based interface to libpfm/perfmon2**

> **Provides convenient access to performance counters but also high level information**

> **Wide range of functionality:**

- Counting events
- Sampling in regular intervals
- Flat profile
- System wide mode
- Triggers
- Different data readout “plug-ins” (modules) available

Perfmon2/pfmon related activities

- **Heavy testing - running on several batch nodes in the background**
  - simple script running in system wide mode (AH, AN)
  - collecting live runtime information about the job characteristics
  - spec jobs also monitored (AH)
- **Constantly submitting patches and fixing bugs**
- **Additional tools:**
  - gpfmon – a graphical frontend to pfmon
  - a wrapper script providing high level information like CPI, L2 misses etc.

Perfmon2/pfmon related activities

- **Training organization**
  - First invitation-only session held last week (14th of March)
  - Went very well
  - Perfmon2 will be highlighted during the Cern School of Computing
  - Future architecture/performance tuning workshops are being planned
    - March 2008 (dry run; by invitation only)
    - Summer 2008 (CSC)
    - Fall 2008 (CERN)
- **Starting a collaboration with PH parallelism R&D project**

Changes in perfmon2 over the last year

- Patch updated to the latest Linux versions
- Extensive additional support for non-x86 architectures (in particular the CELL)
- Numerous conflicts removed (with other patches)
- Patch being updated constantly and being kept “up to date”

Changes in pfmon over the last 4 months

> **Re-assessment of the CERN contribution**

> **200kB patch written from scratch**

- Symbol resolution for all executed code (no matter if exec'd, forked, threaded etc)
- Symbol demangling engine reworked
- Trigger code reworked
- Numerous minor updates, numerous bugfixes
- Updated for compatibility with complex CERN software – ROOT etc
- ~12-15 man weeks coding (5 weeks real time)
  - Andrzej Nowak (CERN) + Stephane Eranian (HP Labs)
- Main part of CERN's contribution finalized, changes committed to CVS, YE2007

> **Additional contributions on CERN's behalf expected**

Mainstream perspectives for perfmon2

- **Inclusion in the mainline kernel is very important for widespread adoption, even at CERN**
- **Status from March: the updated patch is being split up into pieces for review (yet again)**
- **The Linux community created a very demanding work environment**
- **Inclusion in the mainline this quarter is possible, but still a large effort is required**
  - There's not much openlab can do to help

Pfmon usage

Basic modes

> **Counting**

- Example: How many instructions did my application execute?

> **Sampling**

- Reporting results in “regular” intervals

> **Profiling**

- Example: how many cycles are spent in which function?
- Example: how many cache misses occur in which function?

Enabling different modes

> Different modes are triggered by the presence of certain command line switches

> Counting

default mode

> Sampling

```
--smpl-module=compact
```

> Profiling

```
--long-smpl-period=NUM
```

Counting example

1. **Specify interesting events**

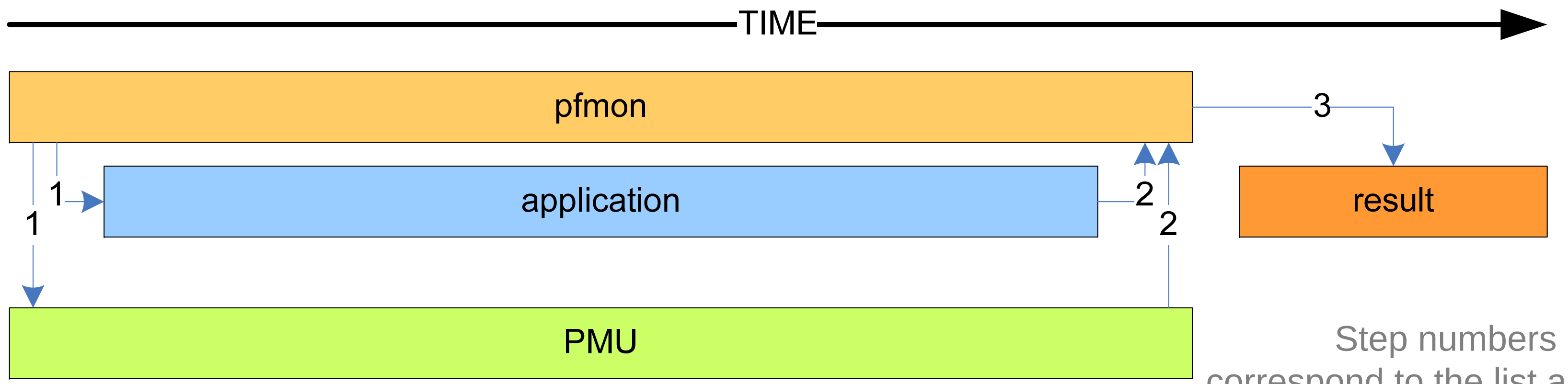
   i.e. INSTRUCTIONS_RETIRED

2. **Build the command line**

   ```
   pfmon -e INSTRUCTIONS_RETIRED ls /xyz
   ```

3. **Run and obtain results**

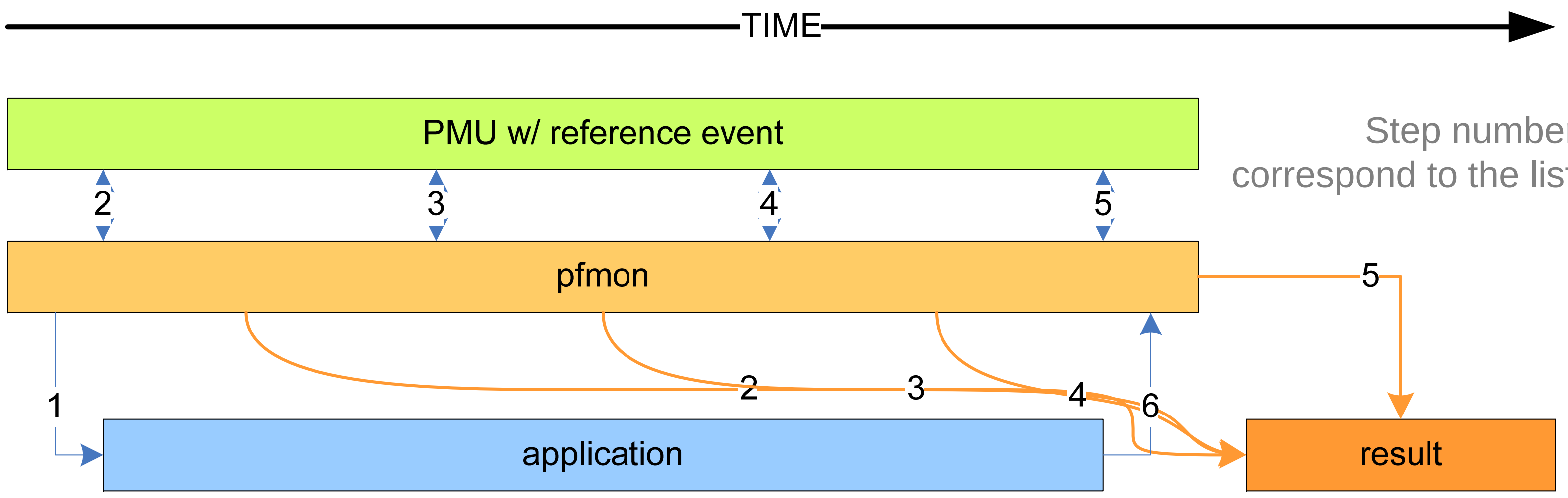
   ```
   181992 INSTRUCTIONS_RETIRED
   ```



Step numbers don't correspond to the list above

Sampling example

1. **Specify interesting events and the reference event**
   i.e. UNHALTED_CORE_CYCLES (ref), INSTRUCTIONS_RETIRED
2. **Build the command line**
   ```
   pfmon -e UNHALTED_CORE_CYCLES,INSTRUCTIONS_RETIRED
      --long-smpl-periods=26670 --smpl-module=compact
      /bin/ls
   ```
3. **Run and obtain results (next page)**



Step numbers don't correspond to the list above

Example sampling results

```
# description of columns:
#      column  1: entry number
#      column  2: process id
#      column  3: thread id
#      column  4: cpu number
#      column  5: instruction pointer
#      column  6: unique timestamp
#      column  7: overflowed PMD index
#      column  8: event set
#      column  9: initial value of overflowed PMD (sampling period)
#      followed by optional sampled PMD values in command line order

1   2     3    4      5                 6                 7 8    9      10
0 32442 32442 2 0x3061230d6a 0x0004d5f49c2a8e57 17 0 -26670 0x556
1 32442 32442 2 0x3061292980 0x0004d5f49c2b4851 17 0 -26670 0xd66
2 32442 32442 2 0x3061226363 0x0004d5f49c2c04dc 17 0 -26670 0x1aaa
3 32442 32442 2 0x3061010159 0x0004d5f49c2c39cb 17 0 -26670 0x6942
4 32442 32442 2 0x306126b5f0 0x0004d5f49c2c9a1c 17 0 -26670 0x171c
```

Profiling example

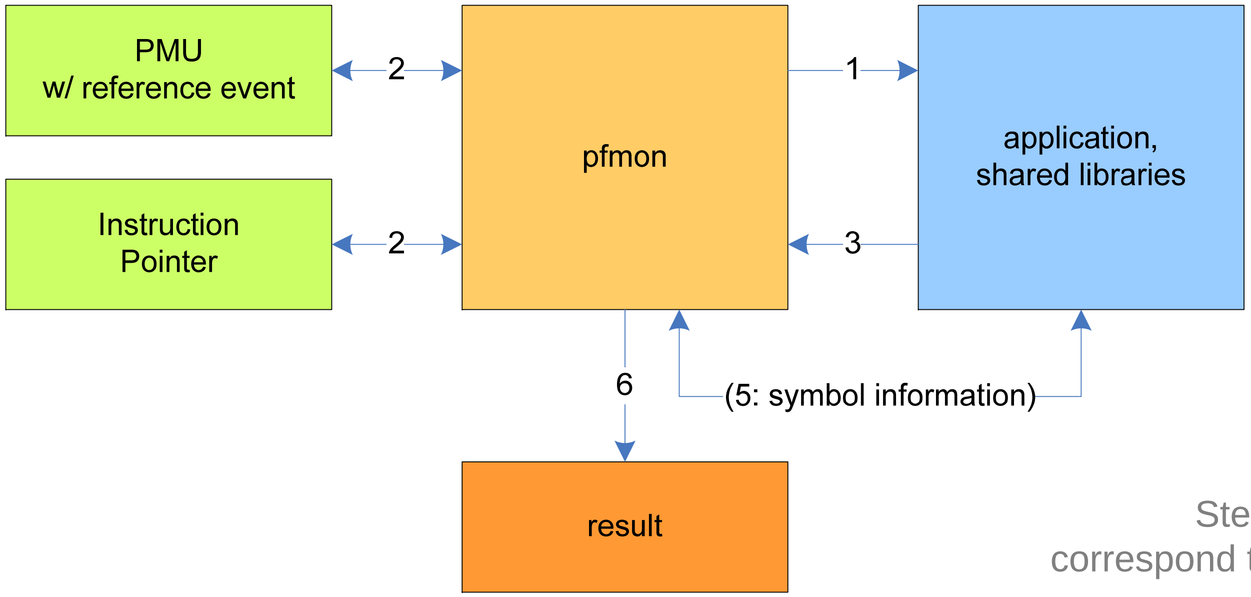
1. **Specify the reference event**

   i.e. UNHALTED_CORE_CYCLES

2. **Build the command line**

```
pfmon -e UNHALTED_CORE_CYCLES --long-smpl-periods=10000
   --resolve-addresses --smpl-per-function /bin/ls
```

3. **Run and obtain results (next page)**



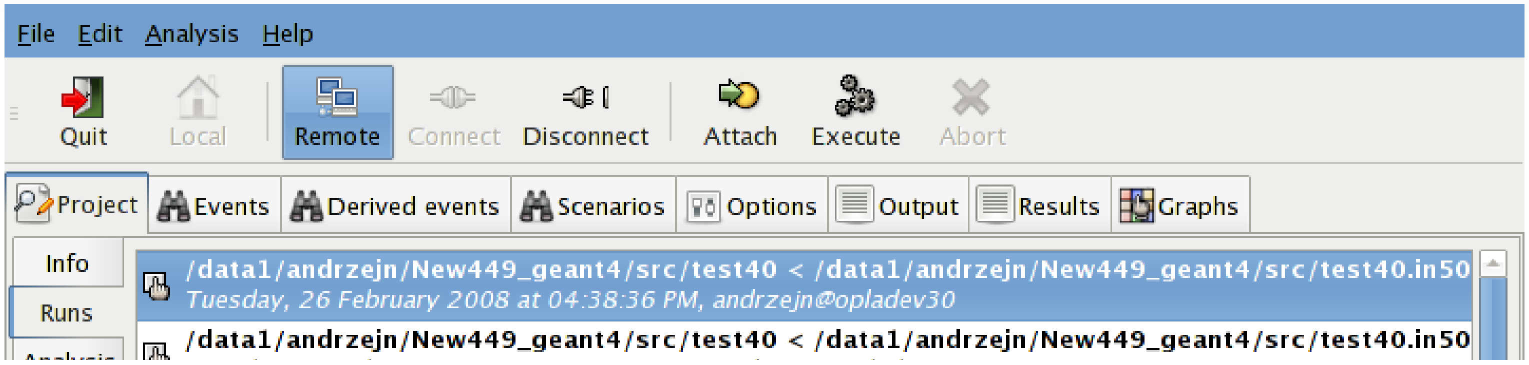
Step numbers don't correspond to the list above

Example profiling results

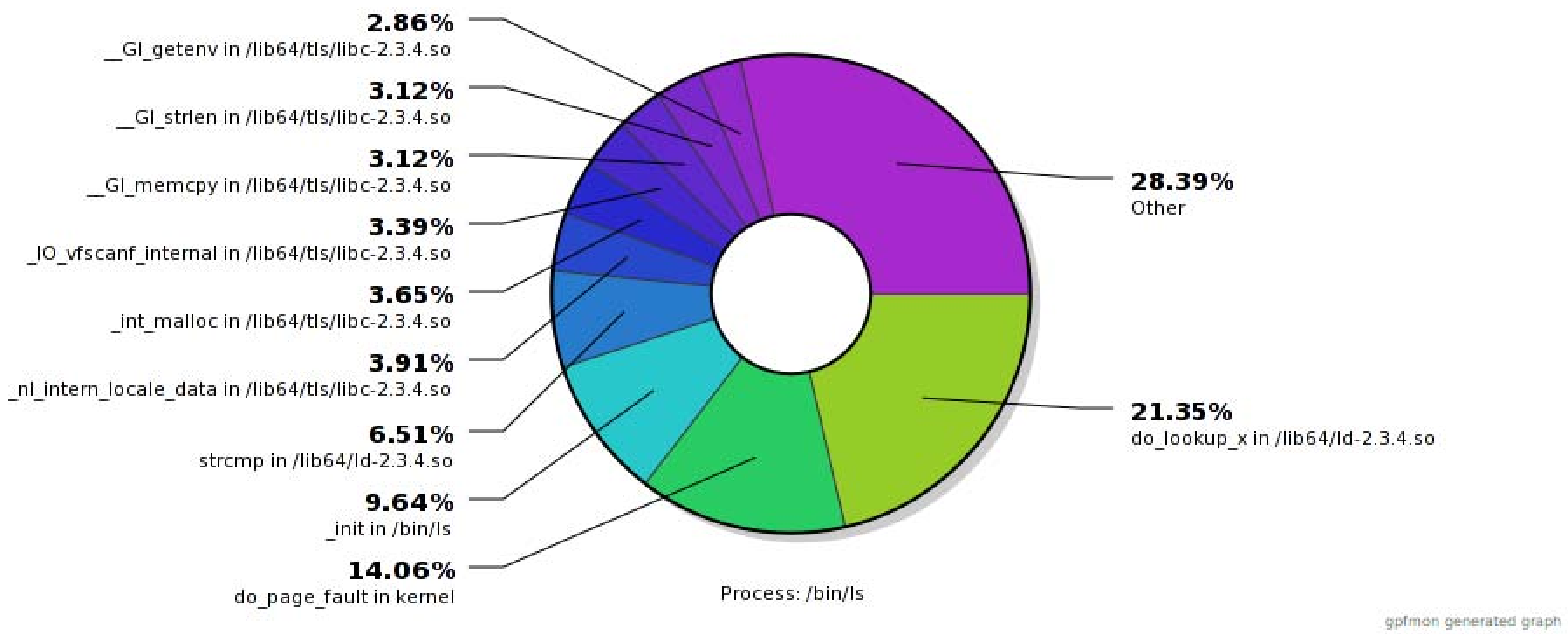
```
cnt  %self   %cum addr symbol
 80 20.83% 20.83% 0x… do_lookup_x</lib64/ld-2.3.4.so>

 53 13.80% 34.64% 0x… do_page_fault<kernel>
 32  8.33% 42.97% 0x… _init</bin/ls>
 20  5.21% 48.18% 0x… __GI_strlen</lib64/tls/libc-2.3.4.so>
 19  4.95% 53.12% 0x… _int_malloc</lib64/tls/libc-2.3.4.so>
 18  4.69% 57.81% 0x… strcmp</lib64/ld-2.3.4.so>
 17  4.43% 62.24% 0x… __GI___strcoll_l</lib64/tls/libc-2.3.4.so>
 13  3.39% 65.62% 0x… __GI_memcpy</lib64/tls/libc-2.3.4.so>
```

gpfmon – a graphical interface for pfmon



/data1/andrzejn/New449_geant4/src/test40 < /data1/andrzejn/New449_geant4/src/test40.in50
Tuesday, 26 February 2008 at 04:38:36 PM, andrzejn@opladev30
/data1/andrzejn/New449_geant4/src/test40 < /data1/andrzejn/New449_geant4/src/test40.in50



Q & A

BACKUP

> Resources:

- http://cern.ch/openlab
- http://sf.net/projects/perfmon2
- http://perfmon2.sourceforge.net (documentation)
- http://perfmon2.sourceforge.net/pfmon_usersguide.html
- http://www.intel.com (manuals)
- http://cern.ch/andrzej.nowak (gpfmon)