Performance monitoring of the software frameworks for LHC experiments

William A. Romero R.
wil-rome@uniandes.edu.co

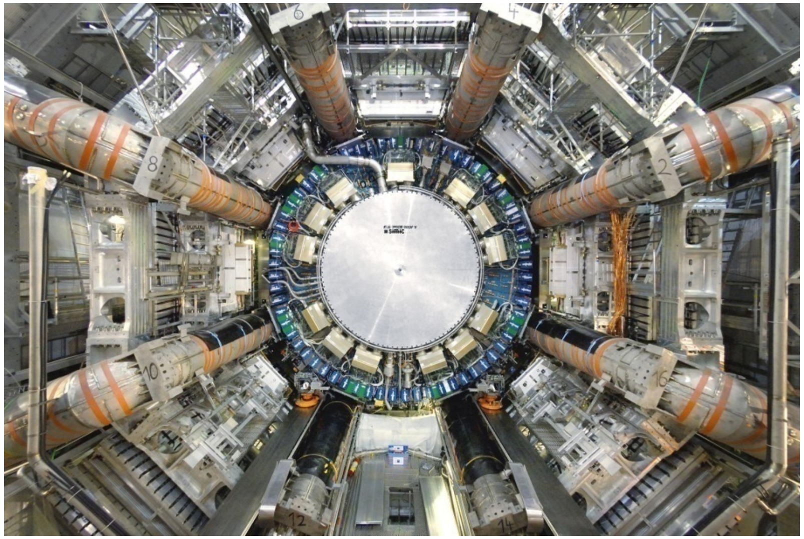
J.M. Dana
Jose.Dana@cern.ch

First EELA-2 Conference
February 27, 2009
Bogotá, COL

OUTLINE

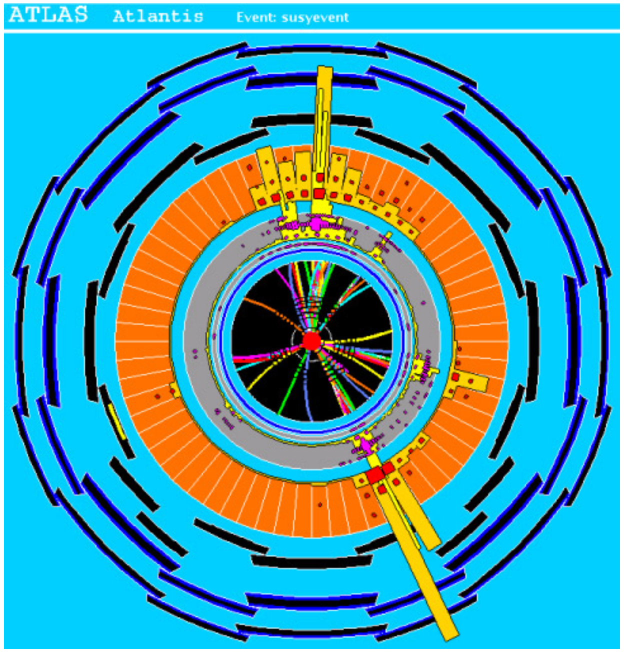
- Introduction
- Performance Monitoring
- Monitoring Tool: PFMON
- *Pfmon deluxe standard* Analysis
- *Pfmon simd1* Analysis
- *Pfmon* profiling
- Application Improvement
- Execution Stages in LHC Software Frameworks
- Monitoring Results
- Concluding Remarks

INTRODUCTION



ATLAS detector [1]

- HEP community has developed huge C++ software frameworks for event generation, detector simulation, and data analysis.



Simulated events [1]

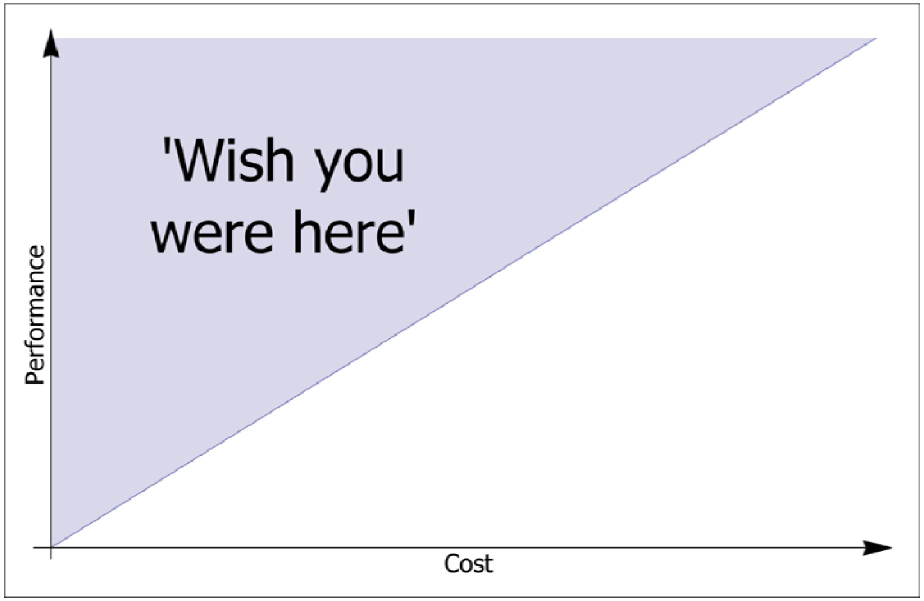
"When LHCb *sees where the antimatter's gone*
ALICE *looks at collisions of lead ions*
CMS & ATLAS *are two of a kind:*
They're looking for whatever new particles they can find
The LHC *accelerates the protons and the lead,*
and the things that it discovers will rock you in the head"

From LHC Rap

PERFORMANCE MONITORING

Save on resources

- $ Costs $
  - Manpower vs. Hardware
- Important power/thermal issues → avoid new hardware additions
- Does speed really matter ?
  → Mandatory!
- LHC physicists have intensive CPU demands



Performance improvements
as a function of cost [2].

PERFORMANCE MONITORING

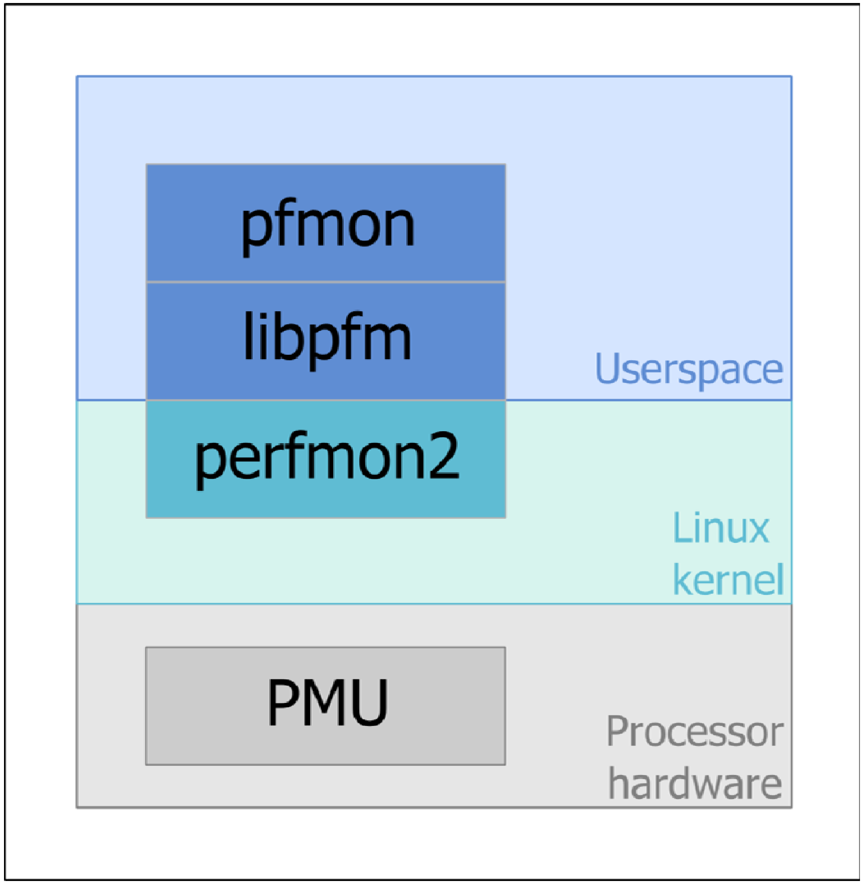
Goal: Identify well-known signs about how the application is being executed:

- Processes (functions/methods)
- Bottlenecks (shared or external libraries)

Performance tuning levels:

- Hardware
- Operating System
- Source code

MONITORING TOOL: PFMON



pfmon components.

- Performance Monitoring Unit (PMU) : a complete and consistent facility in the modern processor architectures.
- ***Perfmon2***: provides a uniform abstract model to access PMU.
  Intel Itanium, Intel P6, P4, P2,
  Pentium M, Core and Core 2,
  AMD Opteron(Dual and Quad-core), etc.
- ***libpfm:*** interface for Perfmon2.
- ***pfmon***: perfmon user client.

PFMON DELUXE STANDARD ANALYSIS

Basic Information:

- Amount of Cycles per Instruction (CPI)
- Percentage of:
  - Memory loads and stores
  - Branch instructions
  - Last-level cache misses
  - Bus utilization
  - Floating point operations
  - Vector operations (SIMD)

CPI	1,0989
Load instructions	45,021%
Store instructions	20,371%
Load & store instructions	65,392%
Resource stalls	48,184%
Branch instructions	14,952%
% of branch instr. mispredicted	2,766%
% of L2 loads missed	1,629%
Bus utilization	4,181%
Data bus utilization	2,510%
Bus not ready	0,450%
Comp. SIMD instr. (newFP)	6,982%
Comp. x87 instr. (oldFP)	0,043%

Pfmon deluxe standard analysis information (ALICE simulation stage)

PFMON DELUXE SIMD1 ANALYSIS

The amount and type of SIMD instructions executed

```
                                    CPI        1,1058
      all computational SIMD instr. 3920435357762
        computational SIMD instr. %        6,885%


     percentages      % of instr    % of comp. SIMD
  SCALAR_SINGLE             3,578%            51,966%
  PACKED_SINGLE             0,000%             0,000%
  SCALAR_DOUBLE             3,307%            48,034%
  PACKED_DOUBLE             0,000%             0,000%
```

Pfmon deluxe simd1 analysis information
(ALICE simulation stage)

PFMON PROFILING

Most frequently visited code address

An insight into program execution

```
# results for [27703<-[27641] tid: 27703]
(/data4/wilrome/gauss/soft/lhcb/GAUSS/GAUSS_v30r5/Sim/Gauss/v30r5/slc4_amd64_gcc34/Gauss.exe
/data4/wilrome/gauss/run/pool_0000/bench.opts)
# total samples          : 64913963
# total buffer overflows : 31696
#
#                   event00
counts  %self %cum    code addr symbol
 27769414.28% 4.28%   0x00002b5c990926c0 CLHEP::RanluxEngine::flat()</data4/wilrome/gauss/soft/lcg/external/clhep
 23658533.64% 7.92%   0x00002b5ca2dcb2e0 G4ElasticHadrNucleusHE::GetLightFq2(int, double)</data4/wilrome/gauss/so
 20660223.18% 11.11%  0x000000306150e370 __ieee754_exp</lib64/tls/libm-2.3.4.so>
 19640963.03% 14.13%  0x0000003061511930 __ieee754_log</lib64/tls/libm-2.3.4.so>
 16226892.50% 16.63%  0x000000306126b5f0 __GI___libc_malloc</lib64/tls/libc-2.3.4.so>
 15088252.32% 18.95%  0x00002b5c9d34e5e0 MagneticFieldSvc::fieldVector(ROOT::Math::PositionVector3D<ROOT::Math::C
 14016872.16% 21.11%  0x0000003061269510 __cfree</lib64/tls/libc-2.3.4.so>
 13450442.07% 23.19%  0x00002b5c9ca8cae0 G4Navigator::LocateGlobalPointAndSetup(CLHEP::Hep3Vector const&, CLHEP::
 11204781.73% 24.91%  0x00000030612695d0 _int_malloc</lib64/tls/libc-2.3.4.so>
```

Results generated by *Pfmon* profiling
(LHCb simulation stage)

APPLICATION IMPROVEMENT

Compiler optimization heuristics are not enough!

Common problems:

- Cache misses
- False sharing
- Excessive floating point operations
- Multi-Core memory bandwidth bottleneck

APPLICATION IMPROVEMENT

Next step

- Isolate an identified class/method, maintaining the code structure.
- Implement a test program to check the class/method execution.

Compiler optimization techniques

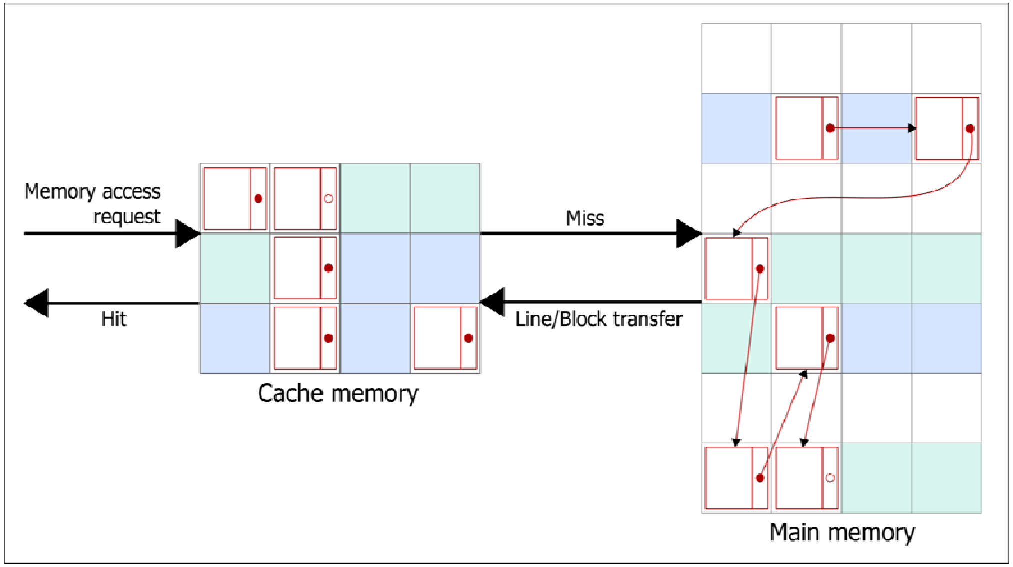
- Intelligent memory management
- Loop unrolling
- Parallelization

APPLICATION IMPROVEMENT

It is necessary previous experience and knowledge about how to map the performance monitoring results into source code improvements.

```
                            Ratios:
                        CPI: 2.0529
         load instructions %: 24.888%
        store instructions %: 14.751%
load and store instructions %: 39.639%
resource stalls % (of cycles): 53.562%
       branch instructions %: 18.223%
     % of branch instr. mispredicted:
                              0.714%
        % of l2 loads missed: 94.554%
            bus utilization %: 8.158%
       data bus utilization %: 4.631%
             bus not ready %: 0.000%
 comp. SIMD instr. ('new FP') %: 1.585%
  comp. x87 instr. ('old FP') %: 0.000%
```

Pfmon results



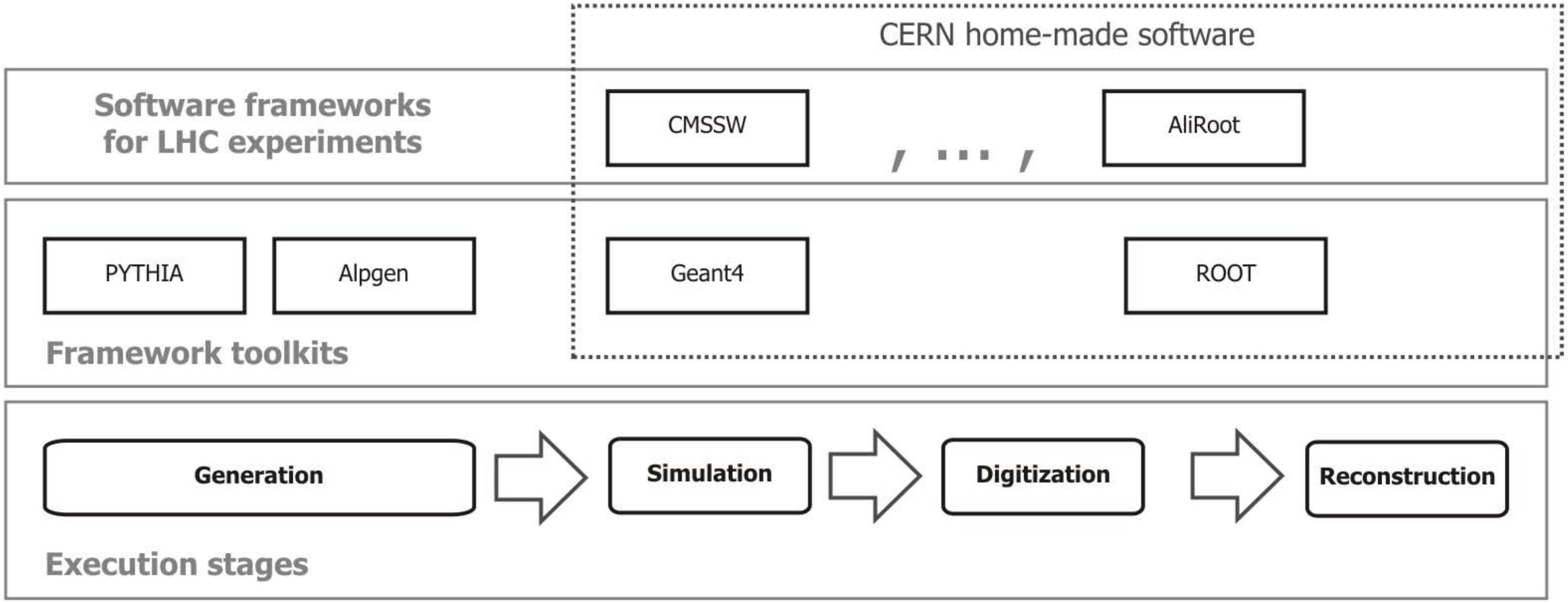
What happens into a cache miss?

PERFORMANCE MONITORING RESULTS

After *pfmon* deluxe analysis and profiling:

- Identified weakness → "the applications symptoms"
- Possible check points into the application code

EXECUTION STAGES IN LHC SOFTWARE FRAMEWORKS



SOME DETAILS

LHCb

- Simulation
- 32-bit & 64-bit

CMS SW

- Generation, Simulation, Digitization & Reconstruction
- 32-bit

ALICE

- Simulation & Reconstruction
- 64-bit

Our testbed

- Intel Xeon Architecture
- 2 processor Dual-Core 2.66 Mhz. 64-bit.
- 4 MB L2 cache
- 8 GB RAM
- Scientific Linux CERN 4.7
- (GCC) 3.4.6 20060404

MONITORING RESULTS - LHCb

n.=events, t:= threads	n:150, t:1	n:150, t:2	n:150, t:4	n:150, t:8
CPI	1,2967	1,298	1,3107	1,3347
Load instructions	36,82%	36,84%	36,82%	36,80%
Store instructions	20,91%	20,94%	20,92%	20,91%
Load & store instructions	57,72%	57,79%	57,74%	57,71%
Resource stalls	26,75%	26,73%	27,61%	28,22%
Branch instructions	14,74%	14,74%	14,72%	14,72%
% of branch instr. mispredicted	3,24%	3,24%	3,25%	3,27%
% of L2 loads missed	0,23%	0,22%	0,39%	0,64%
Bus utilization	**0,73%**	**0,64%**	**2,05%**	**3,25%**
Data bus utilization	0,25%	0,24%	0,76%	1,21%
Bus not ready	0,00%	0,00%	0,00%	0,00%
Comp. SIMD instr. (newFP)	**0,00%**	**0,00%**	**0,00%**	**0,00%**
comp. x87 instr. (oldFP)	**9,66%**	**9,64%**	**9,67%**	**9,67%**

32-bit

64-bit

n.=events, t:= threads	n:150, t:1	n:150, t:2	n:150, t:4	n:150, t:8
CPI	1,4331	1,4388	1,4516	1,4981
Load instructions	31,69%	31,65%	31,61%	31,68%
Store instructions	16,90%	16,87%	16,87%	16,89%
Load & store instructions	48,59%	48,52%	48,48%	48,56%
Resource stalls	30,43%	30,38%	31,51%	32,46%
Branch instructions	15,44%	15,39%	15,39%	15,41%
% of branch instr. mispredicted	3,79%	3,79%	3,83%	3,81%
% of L2 loads missed	0,33%	0,32%	0,54%	0,86%
Bus utilization	**0,77%**	**1,11%**	**3,38%**	**5,19%**
Data bus utilization	0,42%	0,41%	1,26%	1,94%
Bus not ready	0,00%	0,00%	0,00%	0,01%
Comp. SIMD instr. (newFP)	**12,69%**	**12,80%**	**12,78%**	**12,78%**
Comp. X87 instr. (oldFP)	**0,07%**	**0,07%**	**0,07%**	**0,07%**

MONITORING RESULTS - CMS

Unresolved symbols

```
# results for [13234<-[13229] tid: 13267]
   (/data4/wilrome/CMS/SW/slc4_ia32_gcc345/cms/cmssw/CMSSW_2_0_11/bin/slc4_ia32_gcc345/cmsRu
   n
   /data4/wilrome/CMS/SW/slc4_ia32_gcc345/cms/cmssw/CMSSW_2_0_11/bin/slc4_ia32_gcc345/relval
   _main.py)
#
#                    event00
#      counts    %self     %cum            code addr  symbol
        162631    9.18%    9.18% 0x000000004a7a7940  0xf72755c8
         91124    5.14%   14.32% 0x00000000ef487d80 0xf7275598
         77422    4.37%   18.69% 0x000000004a7b1720 0xf726b5b7
         77341    4.37%   23.06% 0x00000000ef4906b0 0xf726b2ba
         75188    4.24%   27.30% 0x00000000f6574b30 0xf7269b87
         71631    4.04%   31.35% 0x00000000f64490a0 0xf7269b83
         51866    2.93%   34.28% 0x00000000f65384c0 0xf7269b50
```

pfmon was never prepared to monitor

32-bit version *dlopen* calls

CONCLUDING REMARKS

- The approach presented was developed upon CERN openlab previous work with *pfmon* as monitoring tool.
- The monitoring methodology: *pfmon* deluxe analysis, *pfmon* profiling and application improvement.
- The results have been sent to software developers in order to let them know about the requirements and bottlenecks of their tools.
- A new functionality has been added to *pfmon* in order to resolve the symbols generated in the profiling for the 32-version of the software

THANKS!

Q & A

CREDITS

[1] The ATLAS Experiment at CERN.
[Online] Available: http://atlas.ch

[2] Behrooz Parhami. Computer Architecture. OXFORD UNIVERSITY PRESS, 2005.