

Perfmon2: A leap forward in Performance Monitoring

**Sverre Jarpe, Ryszard Jurga, Andrzej Nowak
CERN openlab**



**CHEP 2007
6 September 2007**

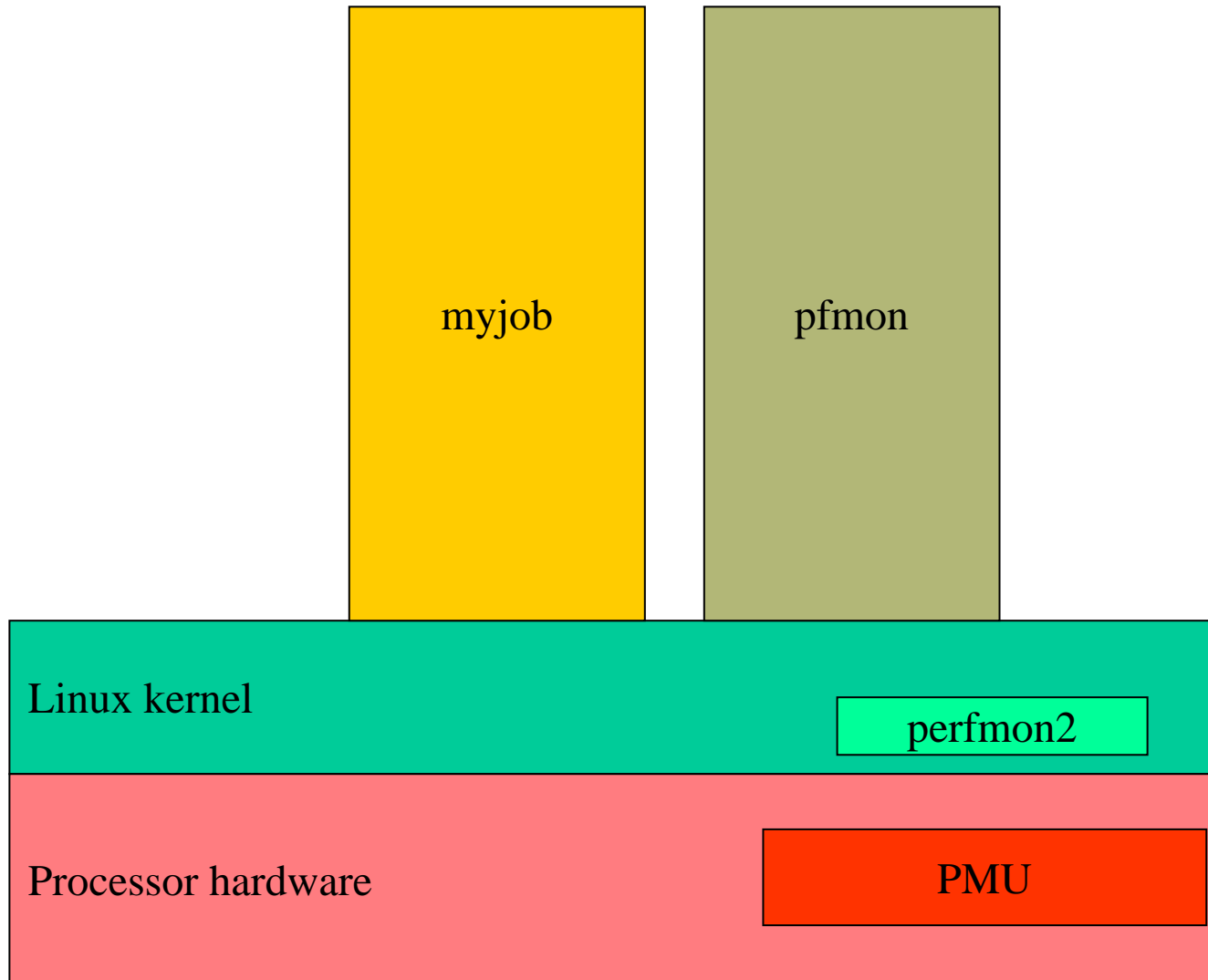


Agenda

- **What is it?**
- **Why do we need it?**
- **The beginning: Software support for the Itanium PMU (Performance Monitoring Unit)**
- **Generalization of software support**
- **Description of perfmon2/pfmon/gpfmon**
- **Examples of usage**
- **Involvement of CERN openlab**
- **Conclusions**



What is where?





Why bother?

- **Is performance analysis worth the effort?**
 - After all, hardware is getting cheaper and cheaper
 - Manpower is usually the expensive part
- **Several justifications might exist, for instance:**
 - The overall cost of hardware is high
 - Hundreds of millions of francs/euros are being spent on the LHC Computing Grid
 - Computer centres are hitting a thermal ceiling
 - Only a fixed number of machines can be accommodated
 - Highly paid people sit and wait for the next answer from the computer
- **And, in addition:**
 - Hardware is getting much more complex
 - Number of cores, execution units, cache levels, etc.

Get yourself a good tool!



Itanium's PMU

- **When the Itanium processor (IA-64 architecture) was developed**
 - A Performance Monitoring Unit (PMU) was made an integral part of the architecture
 - CPU logic for collecting micro-architectural events about:
 - Execution units, caches, buses, etc.
 - Consistent from one processor generation to the next
 - Architected interface between the PMU and the controlling software (operating system)
 - Precise control
 - Region of interest, overflow of counters, etc.
 - Large set of counted events
 - Complete coverage



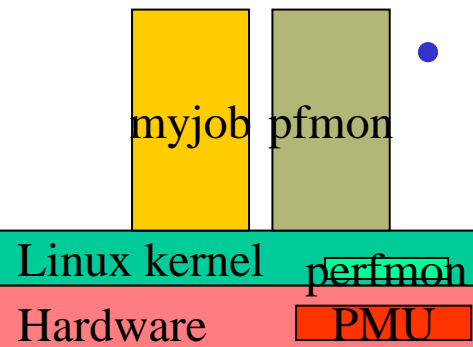
- **Software support initiated already in the early days of Itanium**
 - “Trillian” project ported Linux to IA-64
 - Multiple partners including HP, Intel, and CERN
 - Stéphane Eranian/HP Labs started development of a kernel extension, *perfmon*, for dialoging with the PMU
 - Higher level tools added for ease of use
 - *pfmon*, *q-tools*, etc.



A brilliant idea comes along!

This is why it
is called
“perfmon2”

- **One day Stéphane had an excellent idea:**
 - Expand *perfmon* support to all processors
 - Even if some only have a rudimentary PMU
 - Push for integration into standard Linux kernel
 - No longer a patch that needs to be maintained separately
- **Problem:**
 - His patch touches many sensitive areas in the kernel
 - Context switching, dispatching, etc.
- **Compromise:**
 - Introduce only the infrastructure for *perfmon2* first
 - Over multiple releases
 - Main patch is scheduled for 2.6.23
 - But already now (in 2.6.22) it can be added easily
- **Soon, it will be in the mainstream distributions**
 - Red Hat, SUSE, Ubuntu, etc.
 - And, of course, Scientific Linux





Access to all counters

- Each processor defines a set of counters
 - pfmon -l and pfmon -iMAJORNAME for further details
- Core 2, for instance
 - 129 major counters

UNHALTED_CORE_CYCLES	L2_ADS	L1D_M_EVICT	EXT_SNOOP	BR_CALL_EXEC	MEM_LOAD_RETIRED
INSTRUCTIONS_RETIRED	L2_DBUS_BUSY_RATIO	L1D_PEND_MISS	CMP_SNOOP	BR_CALL_MISSP_EXEC	FP_MMX_TRANS
UNHALTED_REFERENCE_LOADS	L2_LINES_IN	L1D_SPLIT	BUS_HIT_DRV	BR_IND_CALL_EXEC	SIMD_ASSIST
LAST_LEVEL_CACHE_HIT	L2_M_LINES_IN	SSE_PRE_MISS	BUS_HITM_DRV	BR_TKN_BUBBLE_1	SIMD_INSTR_RETIRED
LAST_LEVEL_CACHE_MISS	L2_LINES_OUT	LOAD_HIT_PRE	BUSQ_EMPTY	BR_TKN_BUBBLE_2	SIMD_SAT_INSTR_RETIRED
BRANCH_INSTRUCTION_RETIRED	L2_M_LINES_OUT	L1D_PREFETCH	SNOOP_STALL_DRV	RS_UOPS_DISPATCHED	RAT_STALLS
MISPREDICTED_BRANCHES	L2_IFETCH	BUS_REQUEST_OUTSTANDING	BUS_IO_WAIT	MACRO_INSTS	SEG_RENAME_STALLS
LOAD_BLOCK	L2_LD	BUS_BNR_DRV	L1I_READS	ESP	SEG_REG_RENAMES
SB_DRAIN_CYCLES	L2_ST	BUS_DRDY_CLOCKS	L1I_MISSES	SIMD_UOPS_EXEC	RESOURCE_STALLS
STORE_BLOCK	L2_LOCK	BUS_LOCK_CLOCKS	ITLB	SIMD_SAT_UOP_EXEC	BR_INST_DECODED
MISALIGN_MEM_REF	L2_RQSTS	BUS_DATA_RCV	INST_QUEUE	SIMD_UOP_TYPE_EXEC	BOGUS_BR
SEGMENT_REG_LOADS	L2_REJECT_BUSQ	BUS_TRANS_BRD	CYCLES_L1I_MEM_STALLS	INST_RETIRED	BACLEARS
SSE_PRE_EXEC	L2_NO_REQ	BUS_TRANS_RFO	ILD_STALL	X87_OPS_RETIRED	PREF_RQSTS_UP
DTLB_MISSES	EIST_TRANS	BUS_TRANS_WB	BR_INST_EXEC	UOPS_RETIRED	PREF_RQSTS_DN
MEMORY_DISAMBIGUATION	THERMAL_TRIP	BUS_TRANS_IFETCH	BR_MISSP_EXEC	MACHINE_NUKES	
PAGE_WALKS	CPU_CLK_UNHALTED	BUS_TRANS_INVALID	BR_BAC_MISSP_EXEC	BR_INST_RETIRED	
FP_COMP_OPS_EXE	L1D_CACHE_LD	BUS_TRANS_PWR	BR_CND_EXEC	BR_INST_RETIRED_MISPRED	
FP_ASSIST	L1D_CACHE_ST	BUS_TRANS_P	BR_CND_MISSP_EXEC	CYCLES_INT_MASKED	
MUL	L1D_CACHE_LOCK	BUS_TRANS_IO	BR_IND_EXEC	CYCLES_INT_PENDING_AND_MASKED	
DIV	L1D_ALL_REF	BUS_TRANS_IO	BR_IND_EXEC	SIMD_INST_RETIRED	
CYCLES_DIV_BUSY	L1D_ALL_CACHE_HIT	BUS_TRANS_DEF	BR_IND_MISSP_EXEC	HW_INT_RCV	
IDLE_DURING_DIV	L1D_REPL	BUS_TRANS_BURST	BR_RET_EXEC	ITLB_MISS_RETIRED	
DELAYED_BYPASS	L1D_M_REPL	BUS_TRANS_MEM	BR_RET_MISSP_EXEC	SIMD_COMP_INST_RETIRED	
		BUS_TRANS_ANY	BR_RET_BAC_MISSP_EXEC		



pfmon example (I)

- **Simple counting:**

- pfmon -e

- CPU_CLK_UNHALTED:CORE_P,INST_RETIRED:ANY_P \
 - eu-c --follow-all --no-cmd-output ./stress -b -q

```
337.412 CPU_CLK_UNHALTED:CORE_P /lib/ld-linux.so.2 (32280,32280,32279)
158.874 INST_RETIRED:ANY_P /lib/ld-linux.so.2 (32280,32280,32279)
307.585 CPU_CLK_UNHALTED:CORE_P /lib64/ld-linux-x86-64.so.2 (32281,32281,32279)
173.779 INST_RETIRED:ANY_P /lib64/ld-linux-x86-64.so.2 (32281,32281,32279)
900.025 CPU_CLK_UNHALTED:CORE_P /data1/sverre/rooti51600/test/././stress (32282,32282,32279)
640.584 INST_RETIRED:ANY_P /data1/sverre/rooti51600/test/././stress (32282,32282,32279)
648.442 CPU_CLK_UNHALTED:CORE_P /bin/sh (32282,32282,32279)
501.134 INST_RETIRED:ANY_P /bin/sh (32282,32282,32279)
384.160 CPU_CLK_UNHALTED:CORE_P cat (32283,32283,32279)
109.517 INST_RETIRED:ANY_P cat (32283,32283,32279)
6.584.786 CPU_CLK_UNHALTED:CORE_P /bin/sh (32279,32279,32278)
7.682.214 INST_RETIRED:ANY_P /bin/sh (32279,32279,32278)
1.491.519 CPU_CLK_UNHALTED:CORE_P uname (32285,32285,32278)
577.316 INST_RETIRED:ANY_P uname (32285,32285,32278)
49.115.536.537 CPU_CLK_UNHALTED:CORE_P ./stress (32278,32278,-1)
62.902.094.290 INST_RETIRED:ANY_P ./stress (32278,32278,-1)
```



pfmon example (II)

- **Profiling:**

- `pfmon -e CPU_CLK_UNHALTED:CORE_P --follow-all --smp-epochs-random=0xff:5 --long-smp-epochs=100000 --resolve-addr --demangle-cpp ./stress -b -q`

```
pid=817 (ppid = 811) tid=817 fp=0x3b34d2f8c0 entry_count=1729328
```

```
/data1/andrzejn/tmproot/test/stress
```

```
# counts %self %cum code address
```

```
397530 22.99% 22.99% inflate_fast</data1/sverre/rooti51600/lib/libCore.so>
```

```
217498 12.58% 35.56% deflate_fast</data1/sverre/rooti51600/lib/libCore.so>
```

```
141937 8.21% 43.77% longest_match</data1/sverre/rooti51600/lib/libCore.so>
```

```
136189 7.88% 51.65% TBufferFile::ReadFastArrayDouble32(TStreamerElement</data1/s
```

```
81844 4.73% 56.38% compress_block</data1/sverre/rooti51600/lib/libCore.so>
```

```
62243 3.60% 59.98% Adler32</data1/sverre/rooti51600/lib/libCore.so>
```

```
48143 2.78% 62.76% TBufferFile::ReadFloat</data1/sverre/rooti51600/lib/libRIO.s
```

```
45805 2.65% 65.41% build_tree</data1/sverre/rooti51600/lib/libCore.so>
```

```
42847 2.48% 67.89% TStreamerInfo::ReadBuffer(TBuffer&</data1/sverre/rooti51600/
```

```
27267 1.58% 69.47% inflate</data1/sverre/rooti51600/lib/libCore.so>
```

```
23434 1.36% 70.82% TClonesArray::ExpandCreateFast</data1/sverre/rooti51600/lib/
```

```
23324 1.35% 72.17% __intel_new_memset</data1/andrzejn/tmproot/test/stress>
```

```
22621 1.31% 73.48% inflate_table</data1/sverre/rooti51600/lib/libCore.so>
```

```
17789 1.03% 74.51% TBufferFile::ReadUInt</data1/sverre/rooti51600/lib/libRIO.sc
```



pfmon advantage

**one tool on all
supported
platforms**

# counts	%self	%cum	function name:file
Samples: 901644			
118736	13.17%	13.17%	__ieee754_log:libm-2.3.4.so
85733	9.51%	22.68%	CLHEP::RanecuEngine::flat():libCLHEP-1.9.2.3.so
50836	5.64%	28.32%	__ieee754_exp:libm-2.3.4.so
46250	5.13%	33.45%	G4VProcess::SubtractNumberOfInteractionLengthLeft():ibG4procman.so
31953	3.54%	36.99%	G4SteppingManager::DefinePhysicalStepLength():libG4tracking.so
26342	2.92%	39.91%	G4UniversalFluctuation::SampleFluctuations():libG4emstandard.so
20830	2.31%	42.22%	G4Track::GetVelocity() const:libG4track.so
16984	1.88%	44.10%	cos:libm-2.3.4.so
14004	1.55%	45.66%	G4SteppingManager::InvokePSDIP():libG4tracking.so
13996	1.55%	47.21%	sin:libm-2.3.4.so

Xeon

# counts	%self	%cum	function name:file
Samples: 359161			
41046	11.43%	11.43%	__ieee754_log:/lib64/tls/libm-2.3.4.so
38217	10.64%	22.07%	CLHEP::RanecuEngine::flat():libCLHEP-1.9.2.3.so
24457	6.81%	28.88%	__ieee754_exp:libm-2.3.4.so
16188	4.51%	33.39%	G4UniversalFluctuation::SampleFluctuations():libG4emstandard.so
10620	2.96%	36.34%	G4Track::GetVelocity() const:libG4track.so
10155	2.83%	39.17%	G4VProcess::SubtractNumberOfInteractionLengthLeft():libG4procman.so
8337	2.32%	41.49%	G4UrbanMscModel::ComputeGeomPathLength(double):libG4emstandard.so
7979	2.22%	43.71%	G4SteppingManager::DefinePhysicalStepLength():libG4tracking.so
7558	2.10%	45.82%	G4UrbanMscModel::SampleCosineTheta():libG4emstanc

Core Duo 2

# counts	%self	%cum	function name:file
Samples: 408514			
43914	10.75%	10.75%	__divdf3:libgcc_s-3.4.6-20060404.so.1
32918	8.06%	18.81%	CLHEP::RanecuEngine::flat():libCLHEP-1.9.2.3.so
24958	6.11%	24.92%	__divdi3:libgcc_s-3.4.6-20060404.so.1
16176	3.96%	28.88%	G4SteppingManager::DefinePhysicalStepLength():libG4tracking.so
10846	2.65%	31.53%	exp:libm-2.3.4.so
10776	2.64%	34.17%	sqrt:libm-2.3.4.so
10276	2.52%	36.69%	G4UniversalFluctuation::SampleFluctuations():libG4emstandard.so
10118	2.48%	39.16%	G4SteppingManager::InvokePSDIP():libG4tracking.so
9199	2.25%	41.41%	G4SteppingManager::Stepping():libG4tracking.so
8541	2.09%	43.50%	log:/lib/tls/libm-2.3.4.so

Itanium



Some relevant event ratios

- **From Intel's Optimization manual:**
 - Cycles per instruction (CPI):
 - $\text{cpu_clk_unhalted:core_p}/\text{IRA}$
 - Cycles wasted due to branch mispredictions (%):
 - $100 * \text{resource_stalls:br_miss_clear}/\text{CCU}$
 - Floating-point instructions (%):
 - $100 * \text{fp_comp_ops_exe}/\text{IRA}$
 - Cycles wasted by to FP assists (%):
 - $100 * 80 * \text{fp_assist}/\text{CCU}$
 - L2 cache misses per instruction
 - $\text{l2_lines_in:any}/\text{IRA}$
 - Bus busy (%)
 - $100 * 2 * \text{bus_trans_any:all_agents}/\text{cpu_clk_unhalted:bus}$

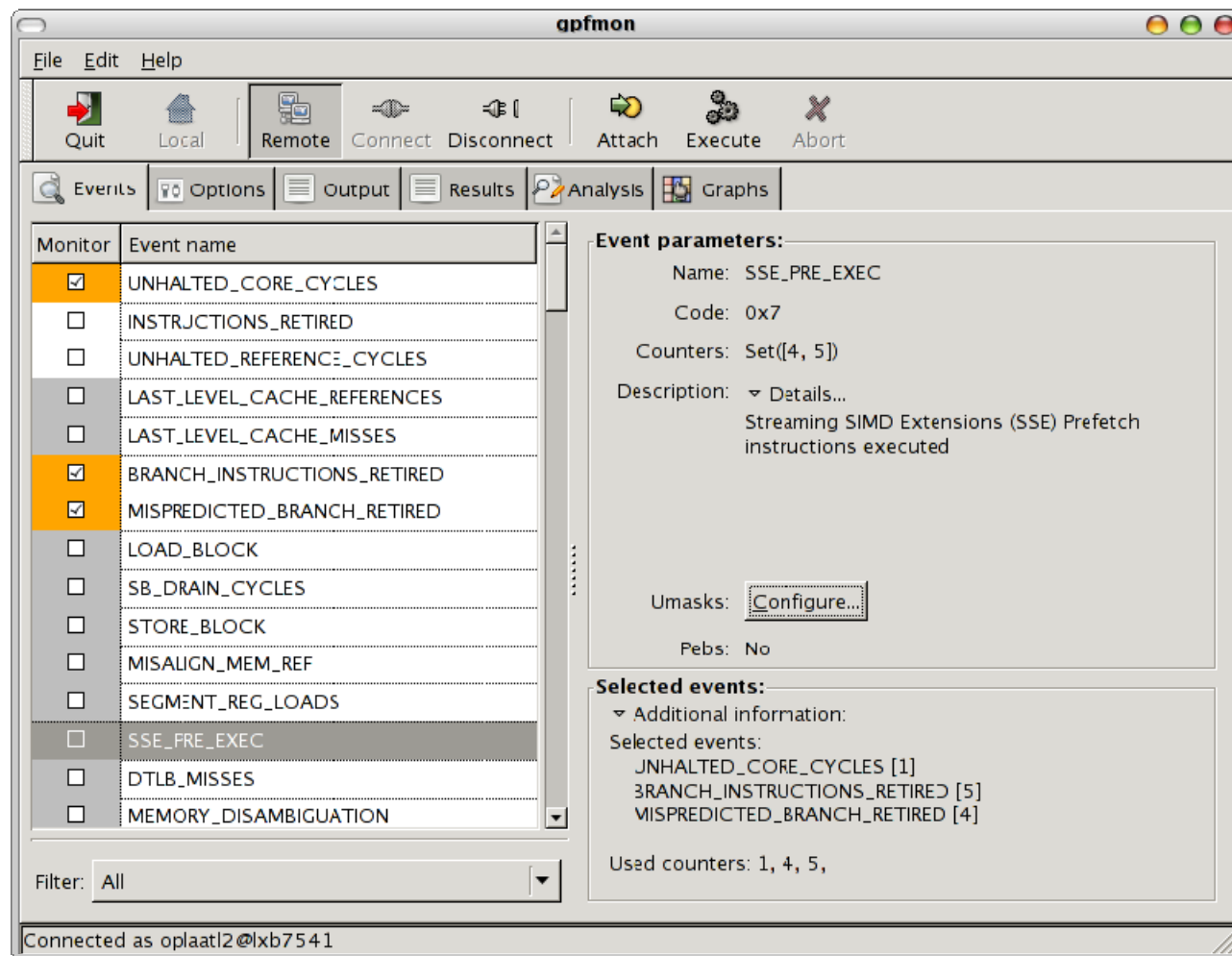
IRA → `inst_retired:any`

CCU → `cpu_clk_unhalted:core_p`



What is *gpfmon* ?

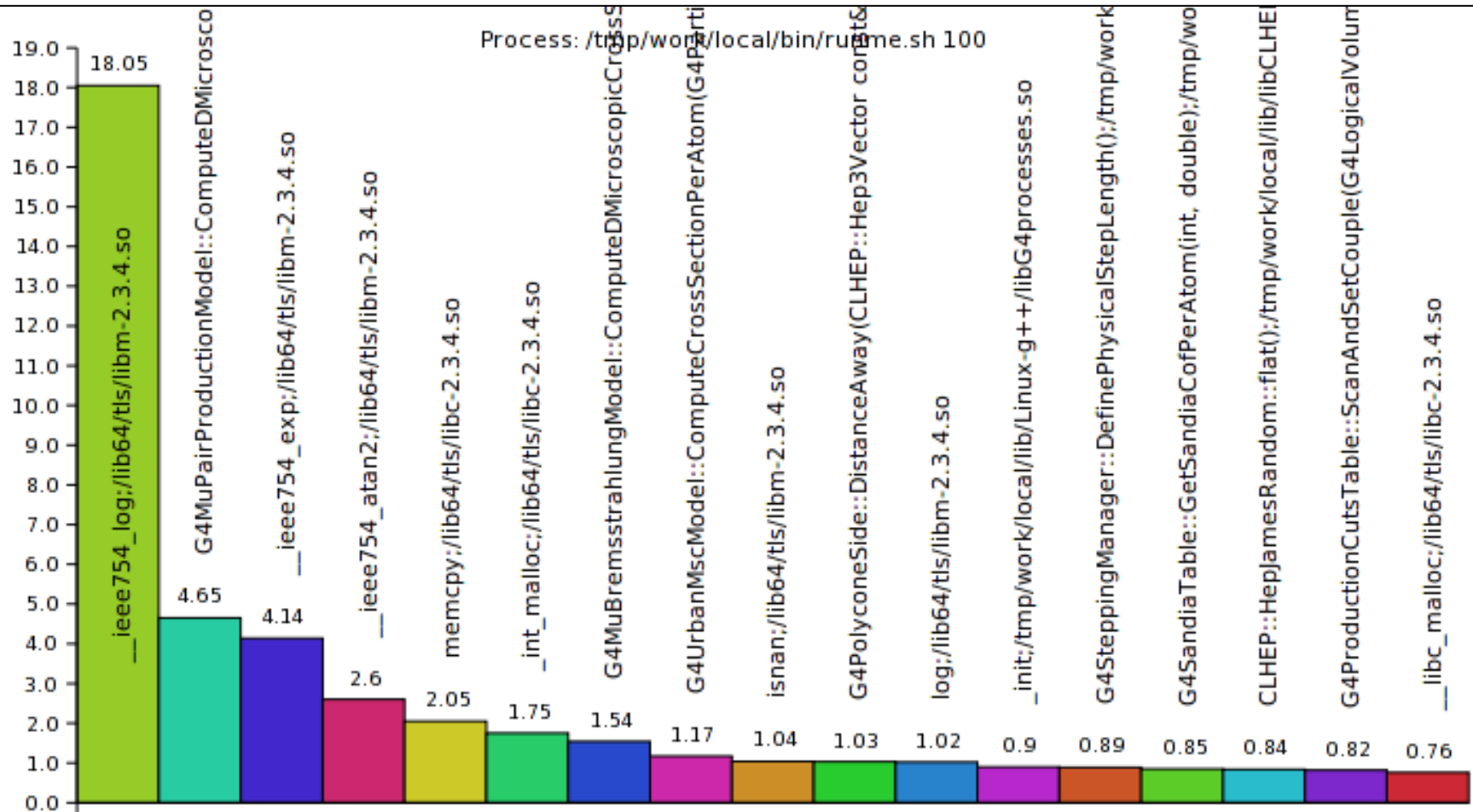
- GUI on top of pfmon:
 - More intuitive approach
 - Still in heavy development
 - More info at: [/cern.ch/andrzej.nowak](http://cern.ch/andrzej.nowak)





Profiling with *gpfmon*

- Example with Geant4 test job:

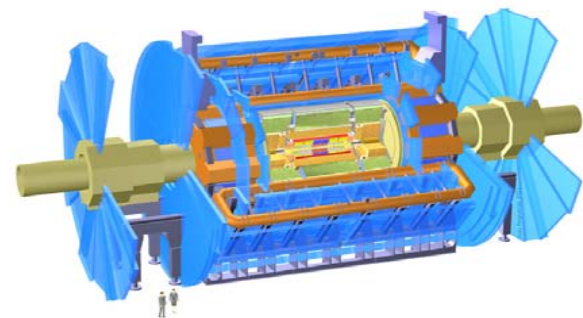


gpfmon generated graph



CERN applications

- **Our frameworks are rather complex**
 - External or internal packages
 - Mix of software languages
 - Dozens (or hundreds!) of shared libraries
 - Auxiliary control via scripts
 - perl, python, etc.
- **Such environments are more difficult to monitor**
 - Inside ATLAS simulation, we observe 400+ dynamic libraries
- **CERN openlab is writing special extensions to *pfmon* to cope with this complexity**
- **Test service established for LHC developers**
 - Available on certain lxbatch machines at CERN





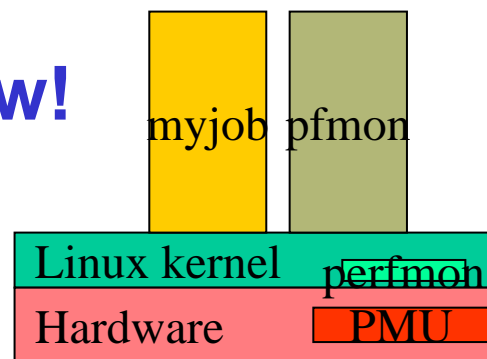
Advanced pfmon options

- ***pfmon* can be used with lots of sophistication:**
 - Kernel/user mode; System-wide
 - Aggregation of results
 - Across multiple threads, processes, or processor cores
 - Multiplexing (when there are not enough “registers” for counting)
 - Sampling (to minimize impact of the monitoring)
 - Triggers
 - On code start/stop or data start/stop (read, write or read/write access)
 - Resolve addresses
 - Map to symbols (when possible)
 - Demangle symbols (C++, Java)
 - Attach to running tasks



Conclusions

- **A ubiquitous approach to performance monitoring is coming:**
 - *perfmon2/pfmon* as well as openlab's *gpfmon*
 - Portable across platforms
- **Access to hundreds of hw counters**
 - All popular processors
 - All mainstream Linux distributions
 - Non-intrusive access for all users
- **Start simple and grow in sophistication over time**
- **Call to action: Get ready now!**





Resources

- perfmon2 home page:
 - <http://perfmon2.sourceforge.net/>
- pfmon tool home page:
 - http://perfmon2.sourceforge.net/pfmon_usersguide.html
- Graphical frontend:
 - <http://cern.ch/andrzej.nowak>
- S.Eranian/HP: “Update on the perfmon2 monitoring interface”
 - http://www.ice.gelato.org/apr06/pres_pdf/gelato_ICE06apr_perfmon_eranian_hp.pdf
- R.Jurga/CERN: “Recent developments in performance monitoring”, openlab Quarterly Review Meeting – 31/01/07
 - http://openlab-mu-internal.web.cern.ch/openlab-mu-internal/Documents/3_Presentations/Presentation_list_2007.htm