The CMS Software Performance at the Start of Data Taking

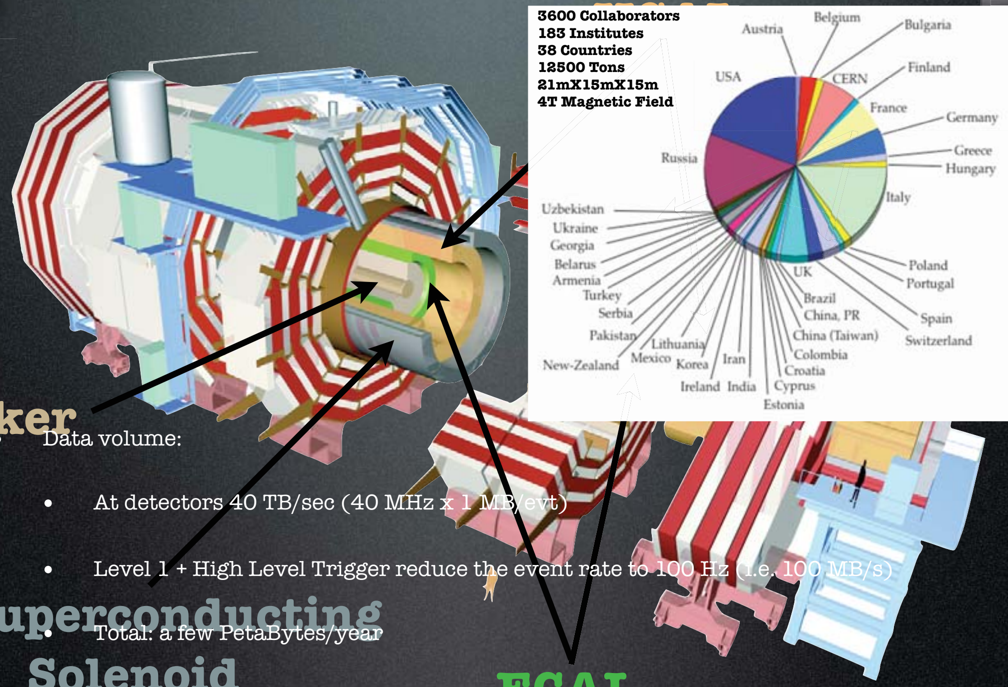
Gabriele Benelli
CERN

Technisches Seminar, DESY Zeuthen, October 21st 2008

Outline

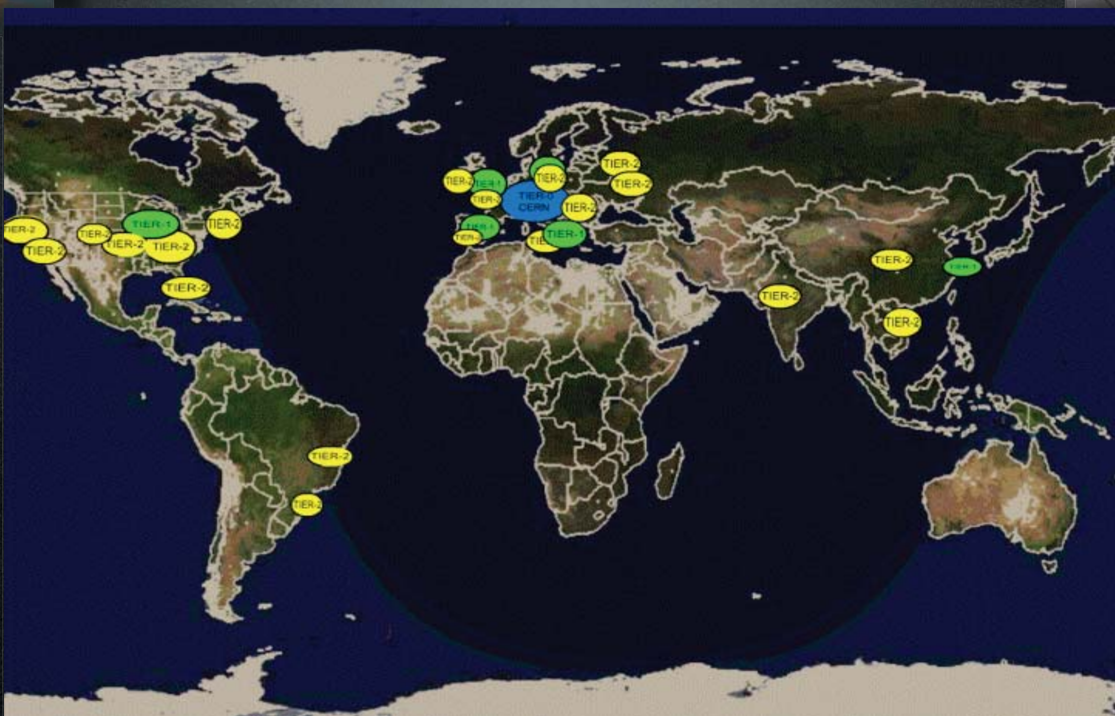
- The CMS Experiment
- CMS Offline software
- Performance Profiling in CMS
- CMSSW Performance Suite
- Using CMSSW as a benchmarking tool
- Conclusions/Outlook

The CMS experiment



- Data volume:
  - At detectors 40 TB/sec (40 MHz x 1 MB/evt)
  - Level 1 + High Level Trigger reduce the event rate to 100 Hz (i.e. 100 MB/s)
  - Total: a few PetaBytes/year
- Offline reconstruction (simulation and analysis) done using the Grid, thousands of computers distributed worldwide, organized in Tier centers

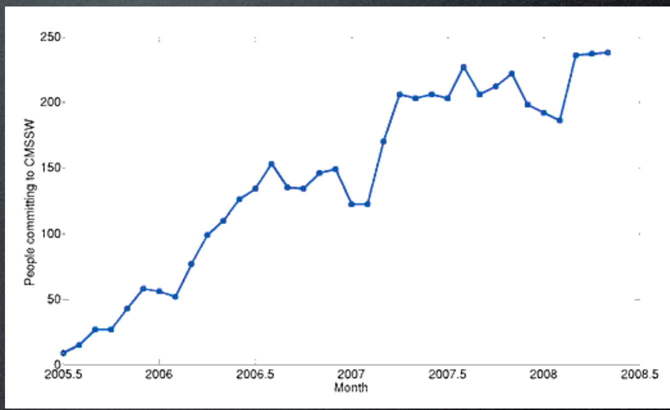
Tiered Computing Model



CMS Offline Software

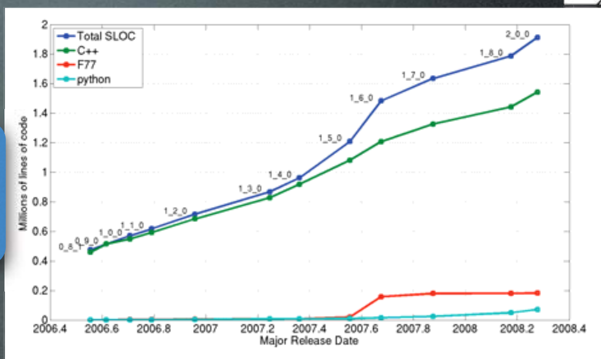
- CMS software framework (CMSSW) consists of approximately:
  - 1100 individual (CVS) packages, organized in 100 Subsystems
  - 2 Millions lines of code (SLOC)
  - 100 external packages (mainly Open Source)
  - 1.5 GB of “data” packages (mainly for Fast Simulation)

250 active developers (a lot of work in the last 3 years!)

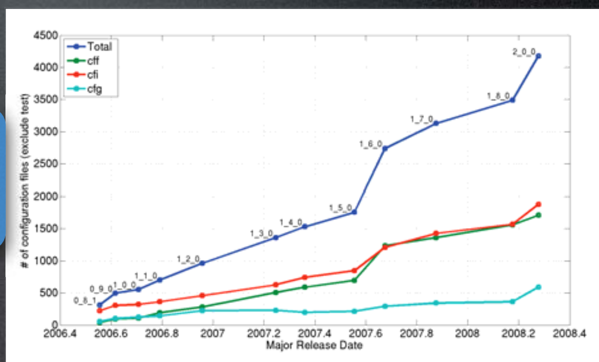


Development metrics

Source Lines Of Code (SLOC) in C++, Fortran, Python
[F77 contribution due to externally developed generators]



Configuration Language contribution

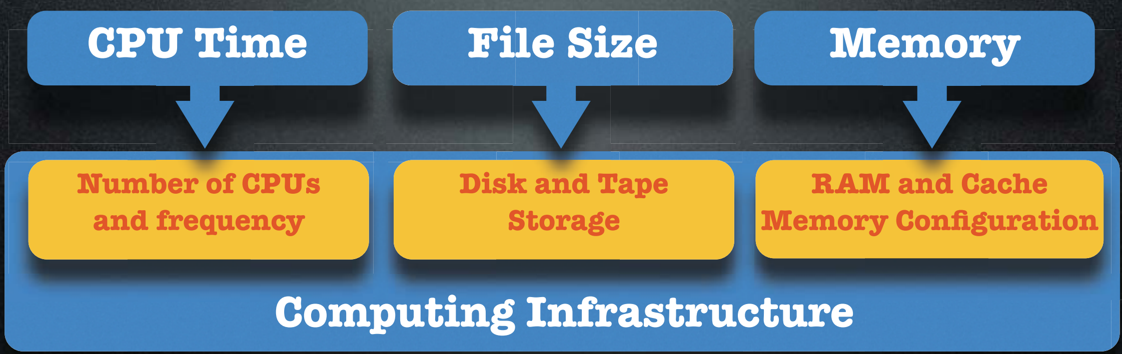


CMS Offline Development Model

- New tags get queued by developers and need to be blessed (i.e. tested) by the relevant level 2 manager for them to be collected by the Tag Collector
- Integration Builds:
  - Two per day for each release cycle and platform
- Development and Production Releases
  - “open” and “close” phases
- Full set of packages has to build always
- Partial releases done later(FWLite, Online)

Performance Profiling

- The CMS Software Framework (**CMSSW**) rapid evolution requires strict quality assurance and the measurement and monitoring of performance is integral part of the Release Validation (**RelVal**) effort
- The 3 key metrics of software performance:



- The monitoring of these offers guidance to software development and optimization effort

Profilers in CMSSW

CPU Time
- Internal: **Timing Service**
- External: **IgProfPerf**, **Callgrind**

File Size
- Internal: **EdmEventSize**

Memory
- Internal: **SimpleMemoryCheck**
- External: **IgProfMem**, **Memcheck**

- CMSSW contains internal tools to profile itself in terms of CPU time, file size and memory use
- Also external tools (IgProf and Valgrind) are used to gain more information, with significant penalty in terms of execution time
- A Performance Suite of tools has been developed to make use of all these profiling tools, integrating them in the Release Validation, providing a regular measurement of CMSSW performance and allowing regression between software releases

The CMSSW Performance Suite

HiggsZZ4LM200
MinBias
SingleElectronE1000
SingleMuMinusPt10
SinglePiMinusE1000
TTbar
QCD_80_120

- Several physics processes (**candles**) have been selected among the ones used in RelVal, in order to probe all aspects of code
- All candles are profiled using the internal profilers, while for the more time consuming external tools specific candles and fewer events are used

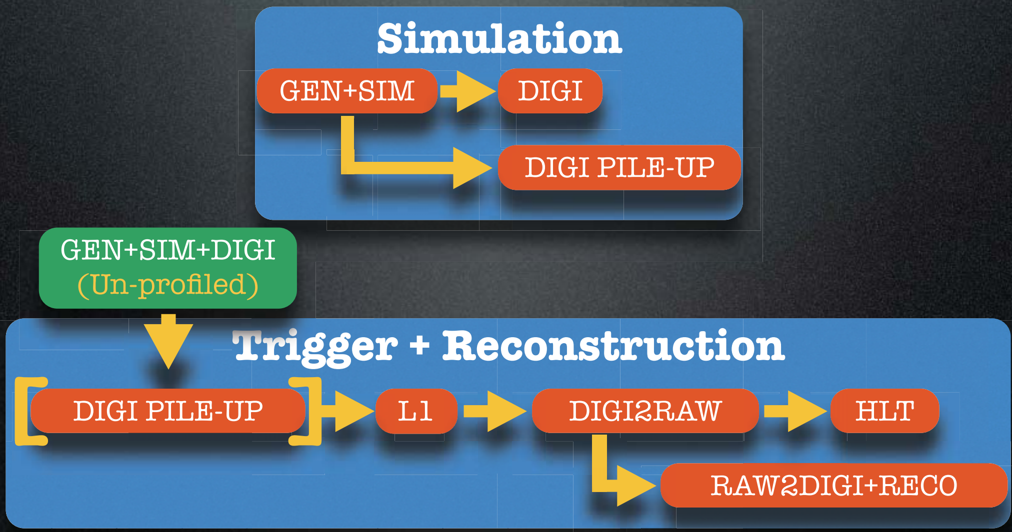
IgProf Profilers
IgProfPerf, IgProfMem(Total), IgProfMem(Live), IgProfMem(Analyse)
5 events/candle

Valgrind Profilers
ValgrindFCE(callgrind), Valgrind(memcheck)
1 event/candle

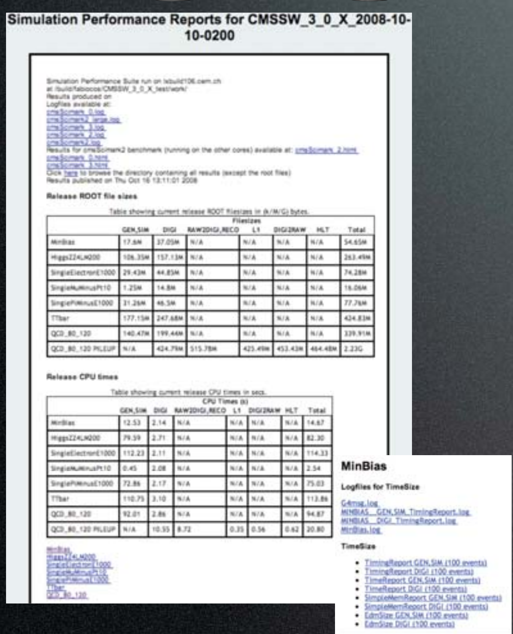
Internal Profilers
TimeReport, Timing Service, EdmSize, SimpleMemoryCheck
100 events/candle

Profiling CMSSW Steps

- Performance profiling is based on simulated data and it is done for each step individually (output of one step used as input for the next)
- CMSSW pre-release development cycle is 1 week, so based on the different steps execution time, two **24hrs**-workflows have been foreseen, running in parallel on separate machines:



CMSSW Performance Suite



Simulation Performance Reports for CMSSW_3_0_X_2008-10-10-0200

Release ROOT file sizes

Table showing current release ROOT filesizes in (k/M/G) bytes.

	Filesizes						
	GEN,SIM	DIGI	RAW2DIGI,RECO	L1	DIGI2RAW	HLT	Total
MinBias	17.6M	37.05M	N/A	N/A	N/A	N/A	54.65M
HiggsZZ4LM200	106.35M	157.13M	N/A	N/A	N/A	N/A	263.49M
SingleElectronE1000	29.43M	44.85M	N/A	N/A	N/A	N/A	74.28M
SingleMuMinusPt10	1.25M	14.8M	N/A	N/A	N/A	N/A	16.06M
SinglePiMinusE1000	31.26M	46.5M	N/A	N/A	N/A	N/A	77.76M
TTbar	177.15M	247.68M	N/A	N/A	N/A	N/A	424.83M
QCD_80_120	140.47M	199.44M	N/A	N/A	N/A	N/A	339.91M
QCD_80_120 PILEUP	N/A	424.79M	515.78M	425.49M	453.43M	464.48M	2.23G

Release CPU times

Table showing current release CPU times in secs.

	CPU Times (s)						
	GEN,SIM	DIGI	RAW2DIGI,RECO	L1	DIGI2RAW	HLT	Total
MinBias	12.53	2.14	N/A	N/A	N/A	N/A	14.67
HiggsZZ4LM200	79.59	2.71	N/A	N/A	N/A	N/A	82.30
SingleElectronE1000	112.23	2.11	N/A	N/A	N/A	N/A	114.33
SingleMuMinusPt10	0.45	2.08	N/A	N/A	N/A	N/A	2.54
SinglePiMinusE1000	72.86	2.17	N/A	N/A	N/A	N/A	75.03
TTbar	110.75	3.10	N/A	N/A	N/A	N/A	113.86
QCD_80_120	92.01	2.86	N/A	N/A	N/A	N/A	94.87
QCD_80_120 PILEUP	N/A	10.55	8.72	0.35	0.56	0.62	20.80

MinBias

Logfiles for TimeSize

G4msg.log
MINBIAS__GEN,SIM_TimingReport.log
MINBIAS__DIGI_TimingReport.log
MinBias.log

TimeSize

- TimingReport GEN,SIM (100 events)
- TimingReport DIGI (100 events)
- TimeReport GEN,SIM (100 events)
- TimeReport DIGI (100 events)
- SimpleMemReport GEN,SIM (100 events)
- SimpleMemReport DIGI (100 events)
- EdmSize GEN,SIM (100 events)
- EdmSize DIGI (100 events)

- In order to ensure meaningfulness to regression analysis, the Performance Suite is run on 2 dedicated multicore machines
- The power-saving BIOS settings and daemons have been disabled and while we run on one core, we run a small cache-contained benchmark on all the other cores, to ensure the same load conditions.
- Once the profiling is done, the suite writes static html reports for each profile and archives a tarball of the working directory on tape
- Finally, all the logs and static html with tables and graphs are published on a web server.

CPU Time Performance

- Sample regression **CMSSW_3_1_X** (G4 9.2 BETA) vs. **CMSSW_2_1_9**

CPU Time regression summary
GEN-SIM, DIGI

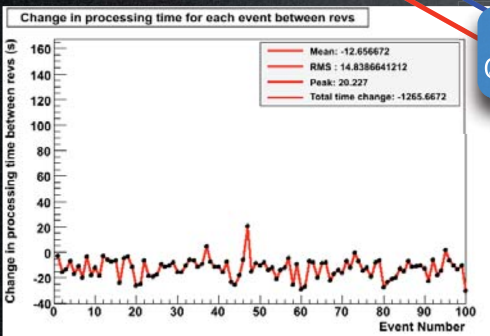
Release CPU times

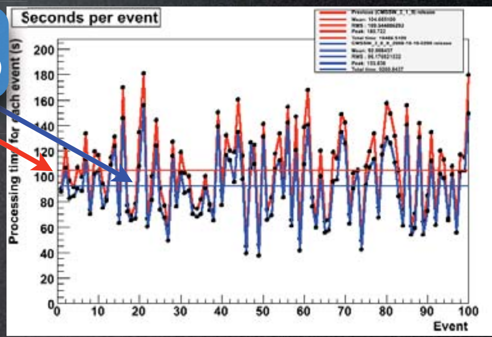
Table showing previous release CPU times, t1, latest times, t2, and the difference between them Δ in secs.

CPU Times (s)

Candles	GEN,SIM t1	GEN,SIM t2	GEN,SIM Δ	DIGI t1	DIGI t2	DIGI Δ	RAW2DIGI,RECO t1	RAW2DIGI,RECO t2	RAW2DIGI,RECO Δ	L1 t1	L1 t2	L1 Δ	DIGI2RAW t1	DIGI2RAW t2	DIGI2RAW Δ	HLT t1	HLT t2	HLT Δ	Total t1	Total t2	Total Δ
MinBias	13.44	12.53	-0.91	0.90	2.14	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.34	14.67	0.33
HiggsZZ4LM200	88.05	79.59	-8.47	1.45	2.71	1.26	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	89.50	82.30	-7.21
SingleElectronE1000	118.09	112.23	-5.86	0.84	2.11	1.27	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	118.92	114.33	-4.59
SingleMuMinusPt10	0.55	0.45	-0.09	0.87	2.08	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.42	2.54	1.12
SinglePiMinusE1000	81.56	72.86	-8.70	0.96	2.17	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	82.52	75.03	-7.49
TTbar	123.36	110.75	-12.61	1.86	3.10	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	125.22	113.86	-11.36
QCD_80_120	104.67	92.01	-12.66	1.62	2.86	1.25	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	106.28	94.87	-11.41
QCD_80_120 PILEUP	N/A	N/A	N/A	8.90	10.55	1.65	32.99	8.72	-24.27	0.35	0.35	0.00	0.54	0.56	0.02	0.61	0.62	0.01	43.39	20.80	-22.59

GEN+SIM
QCD_80_120





CPU Time Performance

- Sample regression **CMSSW_3_1_X** (G4 9.2 BETA) vs. **CMSSW_2_1_9**

CPU Time regression summary
GEN-SIM, **DIGI**

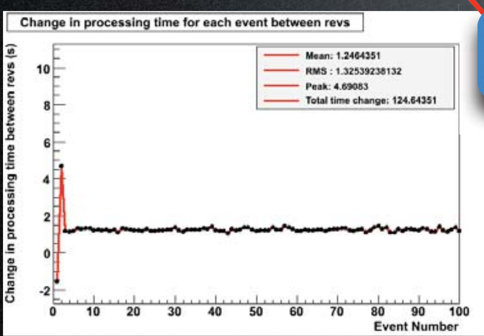
Release CPU times

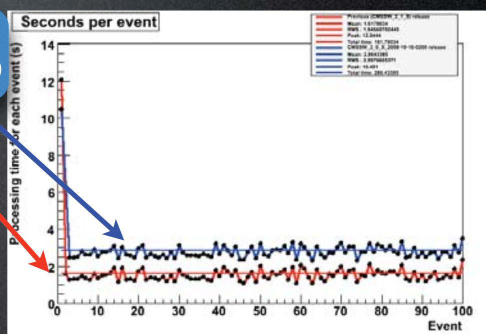
Table showing previous release CPU times, t1, latest times, t2, and the difference between them Δ in secs.

CPU Times (s)

Candles	GEN,SIM t1	GEN,SIM t2	GEN,SIM Δ	DIGI t1	DIGI t2	DIGI Δ	RAW2DIGI,RECO t1	RAW2DIGI,RECO t2	RAW2DIGI,RECO Δ	L1 t1	L1 t2	L1 Δ	DIGI2RAW t1	DIGI2RAW t2	DIGI2RAW Δ	HLT t1	HLT t2	HLT Δ	Total t1	Total t2	Total Δ
MinBias	13.44	12.53	-0.91	0.90	2.14	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.34	14.67	0.33
HiggsZZ4LM200	88.05	79.59	-8.47	1.45	2.71	1.26	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	89.50	82.30	-7.21
SingleElectronE1000	118.09	112.23	-5.86	0.84	2.11	1.27	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	118.92	114.33	-4.59
SingleMuMinusPt10	0.55	0.45	-0.09	0.87	2.08	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.42	2.54	1.12
SinglePiMinusE1000	81.56	72.86	-8.70	0.96	2.17	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	82.52	75.03	-7.49
TTbar	123.36	110.75	-12.61	1.86	3.10	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	125.22	113.86	-11.36
QCD_80_120	104.67	92.01	-12.66	1.62	2.86	1.25	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	106.28	94.87	-11.41
QCD_80_120 PILEUP	N/A	N/A	N/A	8.90	10.55	1.65	32.99	8.72	-24.27	0.35	0.35	0.00	0.54	0.56	0.02	0.61	0.62	0.01	43.39	20.80	-22.59

DIGI
QCD_80_120





CPU Time Performance

- Sample regression **CMSSW_3_1_X** (G4 9.2 BETA) vs. **CMSSW_2_1_9**

CPU Time regression summary
GEN-SIM, **DIGI**

Release CPU times

Table showing previous release CPU times, t1, latest times, t2, and the difference between them Δ in secs.

CPU Times (s)

Candles	GEN,SIM t1	GEN,SIM t2	GEN,SIM Δ	DIGI t1	DIGI t2	DIGI Δ	RAW2DIGI,RECO t1	RAW2DIGI,RECO t2	RAW2DIGI,RECO Δ	L1 t1	L1 t2	L1 Δ	DIGI2RAW t1	DIGI2RAW t2	DIGI2RAW Δ	HLT t1	HLT t2	HLT Δ	Total t1	Total t2	Total Δ
MinBias	13.44	12.53	-0.91	0.90	2.14	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	14.34	14.67	0.33
HiggsZZ4LM200	88.05	79.59	-8.47	1.45	2.71	1.26	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	89.50	82.30	-7.21
SingleElectronE1000	118.09	112.23	-5.86	0.84	2.11	1.27	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	118.92	114.33	-4.59
SingleMuMinusPt10	0.55	0.45	-0.09	0.87	2.08	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.42	2.54	1.12
SinglePiMinusE1000	81.56	72.86	-8.70	0.96	2.17	1.21	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	82.52	75.03	-7.49
TTbar	123.36	110.75	-12.61	1.86	3.10	1.24	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	125.22	113.86	-11.36
QCD_80_120	104.67	92.01	-12.66	1.62	2.86	1.25	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	106.28	94.87	-11.41
QCD_80_120 PILEUP	N/A	N/A	N/A	8.90	10.55	1.65	32.99	8.72	-24.27	0.35	0.35	0.00	0.54	0.56	0.02	0.61	0.62	0.01	43.39	20.80	-22.59

DIGI
QCD_80_120

----------	Modules	in	Path:	digitisation_step
per	event	per	module-visit	
CPU	Real	CPU	Real	Name
0.001190	0.001236	0.001190	0.001236	randomEngineStateProducer
0.142488	0.142527	0.142488	0.142527	mix
0.116912	0.118273	0.116912	0.118273	simSiPixelDigis
0.448742	0.449146	0.448742	0.449146	simSiStripDigis
0.263690	0.264278	0.263690	0.264278	simEcalUnsuppressedDigis
0.066420	0.066442	0.066420	0.066442	simEcalTriggerPrimitiveDigis
0.023786	0.023834	0.023786	0.023834	simEcalDigis
0.000110	0.000154	0.000110	0.000154	simEcalPreshowerDigis
0.078098	0.079732	0.078098	0.079732	simHcalUnsuppressedDigis
0.013418	0.013511	0.013418	0.013511	simHcalTriggerPrimitiveDigis
0.007339	0.007339	0.007339	0.007339	simHcalDigis
0.046983	0.047734	0.046983	0.047734	simMuonCSCDigis
0.019167	0.019201	0.019167	0.019201	simMuonDTDigis
0.000650	0.000675	0.000650	0.000675	simMuonRPCDigis
0.032545	0.032576	0.032545	0.032576	mergedtruth

----------	Modules	in	Path:	digitisation_step
per	event	per	module-visit	
CPU	Real	CPU	Real	Name
0.001100	0.001149	0.001100	0.001149	randomEngineStateProducer
0.144208	0.144232	0.144208	0.144232	mix
1.368812	1.375393	1.368812	1.375393	simSiPixelDigis
0.451491	0.451908	0.451491	0.451908	simSiStripDigis
0.253591	0.254016	0.253591	0.254016	simEcalUnsuppressedDigis
0.067120	0.067173	0.067120	0.067173	simEcalTriggerPrimitiveDigis
0.022927	0.022979	0.022927	0.022979	simEcalDigis
0.000150	0.000154	0.000150	0.000154	simEcalPreshowerDigis
0.077178	0.078801	0.077178	0.078801	simHcalUnsuppressedDigis
0.013438	0.013463	0.013438	0.013463	simHcalTriggerPrimitiveDigis
0.007279	0.007319	0.007279	0.007319	simHcalDigis
0.048613	0.049231	0.048613	0.049231	simMuonCSCDigis
0.018267	0.018247	0.018267	0.018247	simMuonDTDigis
0.000620	0.000618	0.000620	0.000618	simMuonRPCDigis
0.031955	0.032008	0.031955	0.032008	mergedtruth

File Size on disk

- Sample regression **CMSSW_3_1_X** (G4 9.2 BETA) vs. **CMSSW_2_1_9**

CPU Time regression summary
GEN-SIM, DIGI

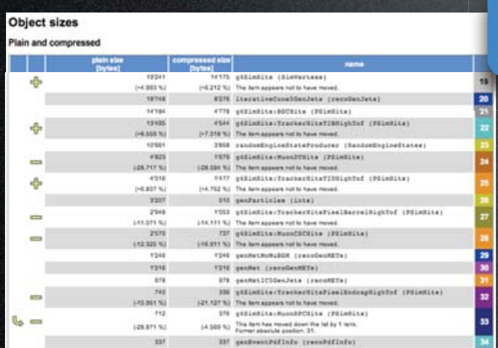
Release ROOT file sizes

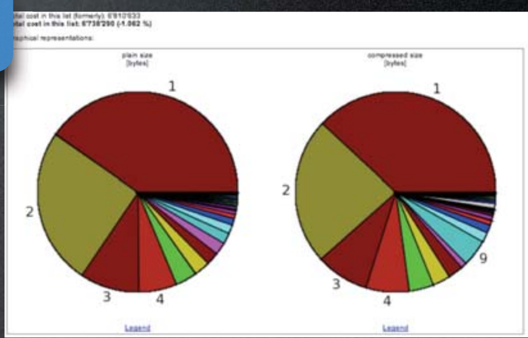
... between them Δ in (k/M/G) bytes.

Filesizes

Candles	GEN,SIM fs1	GEN,SIM fs2	GEN,SIM Δ	DIGI fs1	DIGI fs2	DIGI Δ	RAW2DIGI,RECO fs1	RAW2DIGI,RECO fs2	RAW2DIGI,RECO Δ	L1 fs1	L1 fs2	L1 Δ	DIGI2RAW fs1	DIGI2RAW fs2	DIGI2RAW Δ	fs1
MinBias	17.03M	17.6M	582.11k	36.42M	37.05M	644.79k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
HiggsZZ4LM200	105.88M	106.35M	485.58k	156.26M	157.13M	892.09k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SingleElectronE1000	28.88M	29.43M	567.13k	44.15M	44.85M	717.75k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SingleMuMinusPt10	1.3M	1.25M	-48.6k	14.86M	14.8M	-57.43k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SinglePiMinusE1000	31.78M	31.26M	-532.05k	47.26M	46.5M	-772.32k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TTbar	178.81M	177.15M	-1.66M	249.35M	247.68M	-1.67M	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
QCD_80_120	140.43M	140.47M	38.78k	200.36M	199.44M	-938.8k	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
QCD_80_120 PILEUP	N/A	N/A	N/A	422.9M	424.79M	1.89M	460.51M	515.78M	55.27M	423.62M	425.49M	1.87M	451.27M	453.43M	2.16M	463.54M

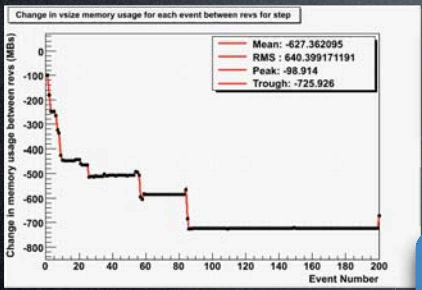
GEN+SIM
TTbar

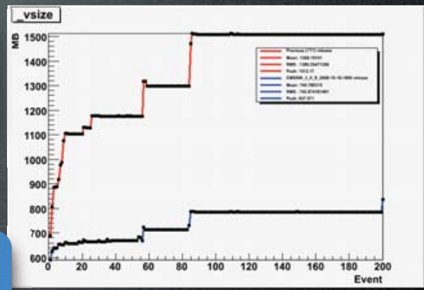




Memory Performance

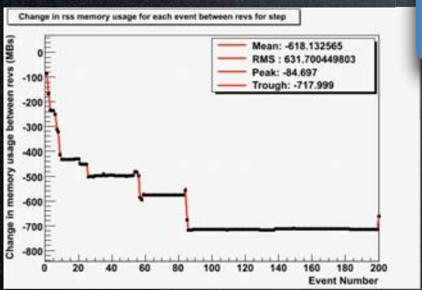
- Virtual memory size (VSIZE) is the parameter constrained in the CMS Computing TDR (1 GB/core).
- The Performance Suite can be used for regression to validate bug fixes with an arbitrary number of events

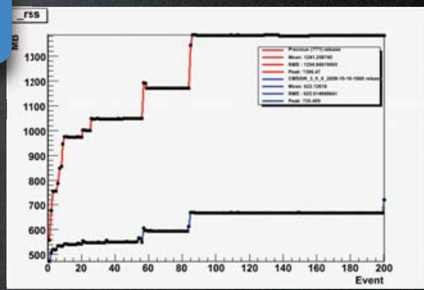




GEN+SIM
TTbar
VSIZE

CMSSW_1_7_1
SimTrack
Fix



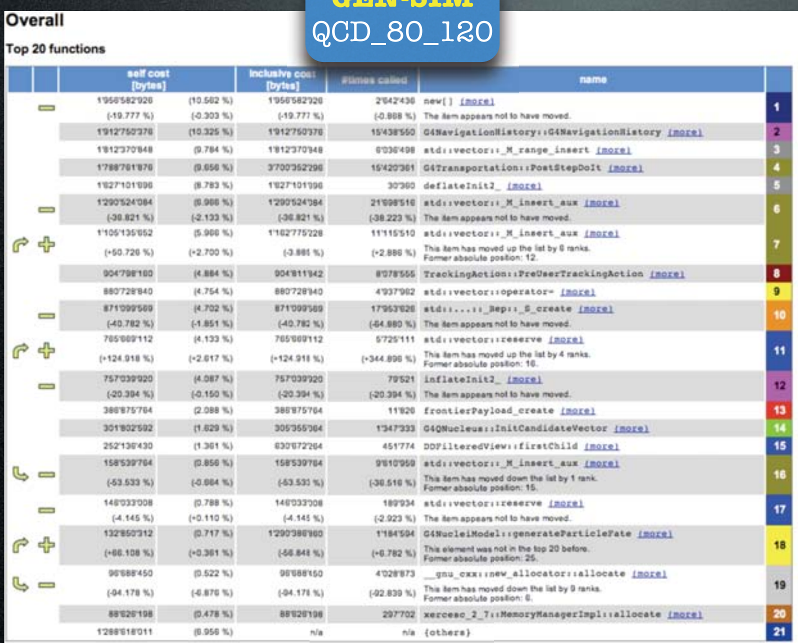


GEN+SIM
TTbar
RSS

Memory Allocation

- IgProf can profile memory allocation, preserving all the callstack information for each function, producing very insightful reports:

GEN-SIM
QCD_80_120

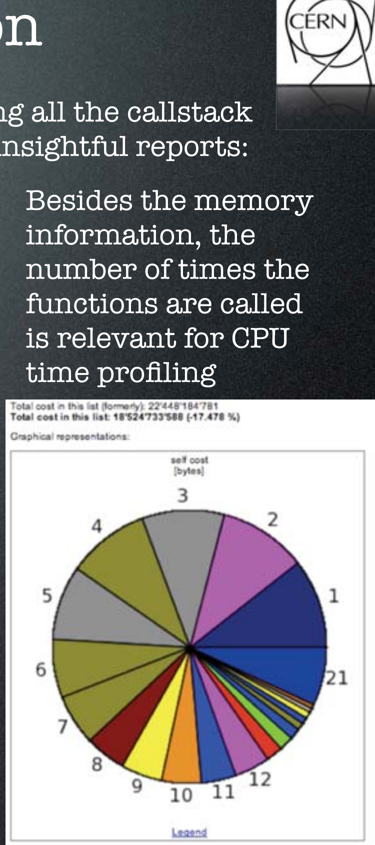


Overall

Top 20 functions

self cost [bytes]		inclusive cost [bytes]	#times called	name	
1'956'582'920	(10.582 %)	1'956'582'920	2'642'436	new[] (more)	1
(-19.777 %)	(-0.303 %)	(-19.777 %)	(-0.888 %)	The item appears not to have moved.	
1'912'750'376	(10.325 %)	1'912'750'376	15'438'550	G4NavigationHistory::G4NavigationHistory (more)	2
1'812'370'848	(9.784 %)	1'812'370'848	6'036'498	std::vector::_M_range_insert (more)	3
1'788'761'876	(9.656 %)	3'700'352'296	15'420'361	G4Transportation::PostStepDoIt (more)	4
1'627'101'096	(8.783 %)	1'627'101'096	30'360	deflateInit2_ (more)	5
1'290'524'084	(6.966 %)	1'290'524'084	21'698'516	std::vector::_M_insert_aux (more)	6
(-38.821 %)	(-2.133 %)	(-36.821 %)	(-38.223 %)	The item appears not to have moved.	
1'105'135'652	(5.966 %)	1'162'775'228	11'115'510	std::vector::_M_insert_aux (more)	7
(+50.726 %)	(+2.700 %)	(-3.881 %)	(+2.886 %)	This item has moved up the list by 6 ranks. Former absolute position: 12.	
904'798'160	(4.884 %)	904'811'942	8'078'555	TrackingAction::PreUserTrackingAction (more)	8
880'728'840	(4.754 %)	880'728'840	4'937'982	std::vector::operator= (more)	9
871'099'569	(4.702 %)	871'099'569	17'953'626	std::...::_Rep::_S_create (more)	10
(-40.782 %)	(-1.851 %)	(-40.782 %)	(-64.880 %)	The item appears not to have moved.	
765'669'112	(4.133 %)	765'669'112	5'725'111	std::vector::reserve (more)	11
(+124.918 %)	(+2.617 %)	(+124.918 %)	(+344.898 %)	This item has moved up the list by 4 ranks. Former absolute position: 16.	
757'039'920	(4.087 %)	757'039'920	79'521	inflateInit2_ (more)	12
(-20.394 %)	(-0.150 %)	(-20.394 %)	(-20.394 %)	The item appears not to have moved.	
386'875'764	(2.088 %)	386'875'764	11'820	frontierPayload_create (more)	13
301'802'592	(1.629 %)	305'355'064	1'347'333	G4QNucleus::InitCandidateVector (more)	14
252'136'430	(1.361 %)	630'672'264	451'774	DDFilteredView::firstChild (more)	15
158'539'764	(0.856 %)	158'539'764	9'810'959	std::vector::_M_insert_aux (more)	16
(-53.533 %)	(-0.664 %)	(-53.531 %)	(-36.516 %)	This item has moved down the list by 1 rank. Former absolute position: 15.	
146'033'008	(0.788 %)	146'033'008	189'934	std::vector::reserve (more)	17
(-4.145 %)	(+0.110 %)	(-4.145 %)	(-2.923 %)	The item appears not to have moved.	
132'850'312	(0.717 %)	1'290'386'960	1'184'594	G4NucleiModel::generateParticleFate (more)	18
(+66.108 %)	(+0.361 %)	(-56.848 %)	(+6.782 %)	This element was not in the top 20 before. Former absolute position: 25.	
96'688'450	(0.522 %)	96'688'150	4'028'873	__gnu_cxx::new_allocator::allocate (more)	19
(-94.178 %)	(-6.876 %)	(-94.178 %)	(-92.839 %)	This item has moved down the list by 9 ranks. Former absolute position: 6.	
88'626'198	(0.478 %)	88'626'198	297'702	xercesc_2_7::MemoryManagerImpl::allocate (more)	20
1'288'618'011	(6.956 %)	n/a	n/a	{others}	21

- Besides the memory information, the number of times the functions are called is relevant for CPU time profiling



Memory Leaks

- Memory errors and leaks are hunted down with Valgrind MemCheck:

CMSSW

as

a machine benchmarking tool

CMSSW benchmarking

- In the context of the HEPiX CPU performance working group CMSSW has been used as a tool to do CPU benchmarking
- The working group charge is to:
  - Test the validity of the industry-standard benchmarks (SPEC CPU) when compared with HEP experiments code
  - Provide some recommendation to guide institutional purchases
- CMSSW applications have been used to benchmark a number of machines with different architecture

CMSSW benchmarking

- All tests run using the Performance Suite, in CMSSW_2_0_0_pre5 release (making use of the CPU time profiling capability)
- All seven "candles" (physics processes) were used
- Run 100 events per candle
- Run GEN+SIM, DIGI, RECO steps separately
- Run the 7 candles sequentially on each core
- Four different tests:
  - Loading all cores simultaneously
  - Loading 1, 3 (only for 4 cores machines) and 5 cores (only for 8 cores machines) with our application while running a cpu-intensive, cache-contained benchmarking tool (cmsScimark2) on the other cores

Benchmarked machines

- Used 10 machines with different architectures, frequencies, memory:
  - 7 machines at CERN from the lxbench cluster
    - Cluster TWiki: https://twiki.cern.ch/twiki/bin/view/FIOgroup/TsiLxbench
  - 1 machine at DESY Zeuten (hpbl1)
  - 2 machines at INFN Padua (lxcmssrv7, lxcmssrv8)

	lxbench01	lxbench02	lxbench03	lxbench04	lxbench05	lxbench06	lxbench07	lxcmssrv07	lxcmssrv08	hpbl1
Number of cores	2	2	4	4	4	4	8	8	8	8
Frequency (GHz)	2.8	2.8	2.2	2.66	3.0	2.6	2.33	2.33	2.1	2.83
Cache (could be L2/L3) (MB)	1	2	2	4	4	2	8	12	2	12
Memory (GB)	2	4	2	8	8	8	16	16	16	16
Processor	Nocona	Irvingdale	Opteron 275	Woodcrest	Woodcrest	Opteron 2218 Rev.F	Clovertown	Xeon HarperTown E5410	Opteron Barcelona 2352	Xeon HarperTown E5440
Vendor	Intel	Intel	AMD	Intel	Intel	AMD	Intel	Intel	AMD	Intel

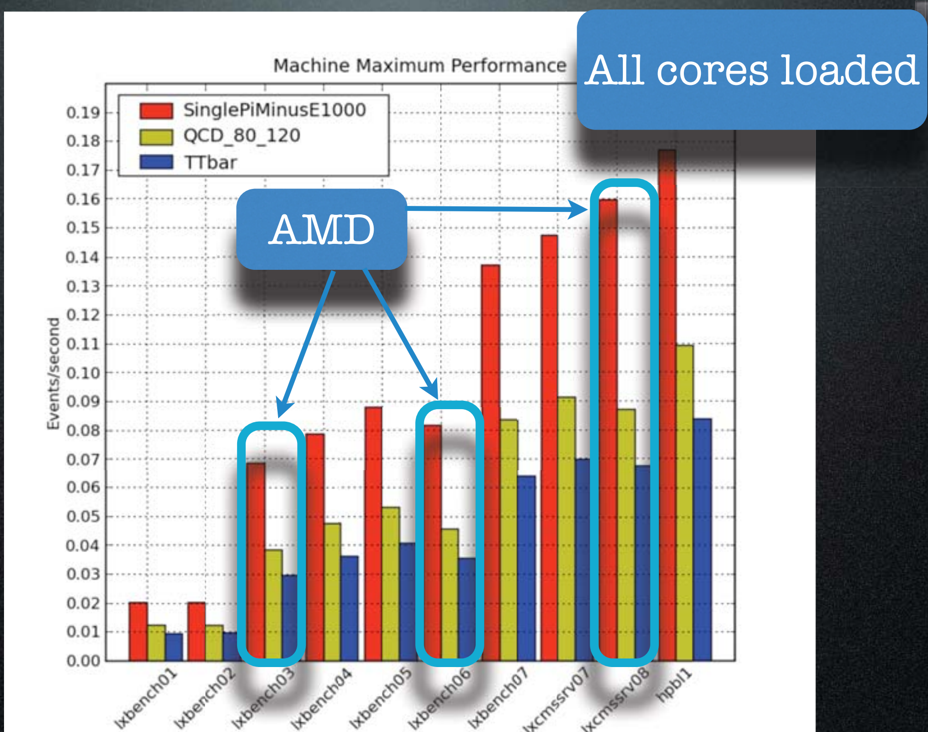
Benchmarking the cluster

- Basically the Performance Suite was submitted on the wanted cores with 100 events and all the internal profilers
- Once the Suite was done running, the logfiles were “harvested” and the timing information from the framework was used to calculate the average for each candle
- The data was collected in a Python dictionary and then tables for publication on the HEPiX Wiki were produced
- Using the same data structure, comparison/analysis plots for the various machines could be generated with MatplotLib

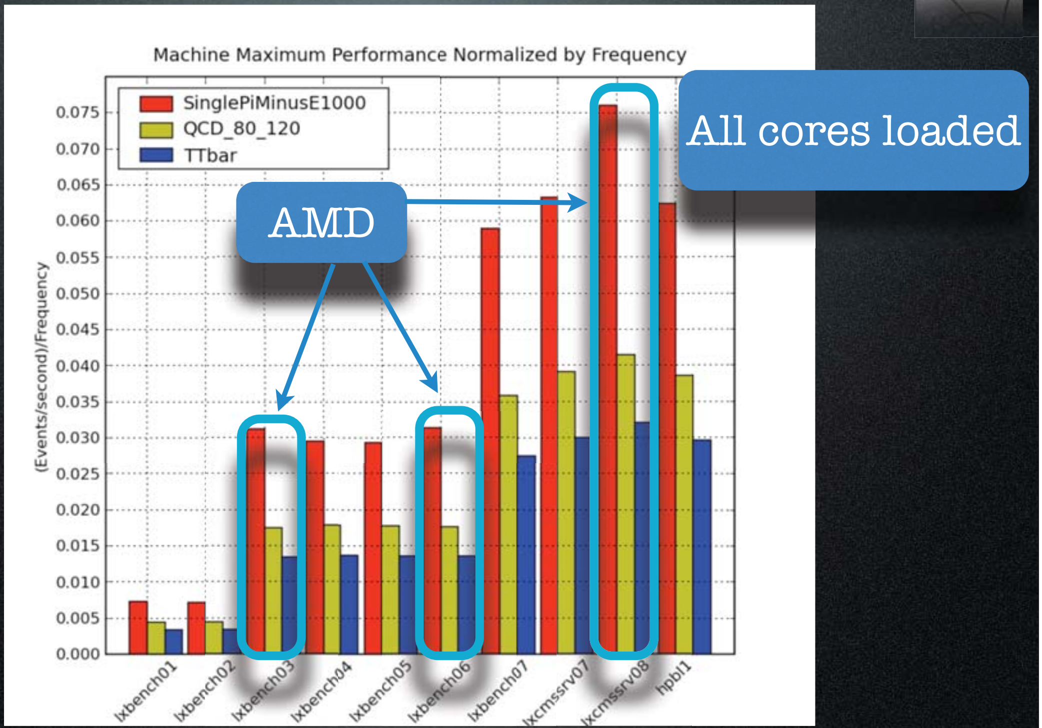
CMSSW Benchmarking

- The result of the benchmarking is seconds/event averaged on the 99 events (skipping the first one to avoid biases due to initialization)
- The results are reported in 3 formats:
    - seconds/event per core
    - events/second per core
    - events/second per machine
- Link: https://hepix.caspur.it/processors/dokuwiki/doku.php?id=benchmarks:cms

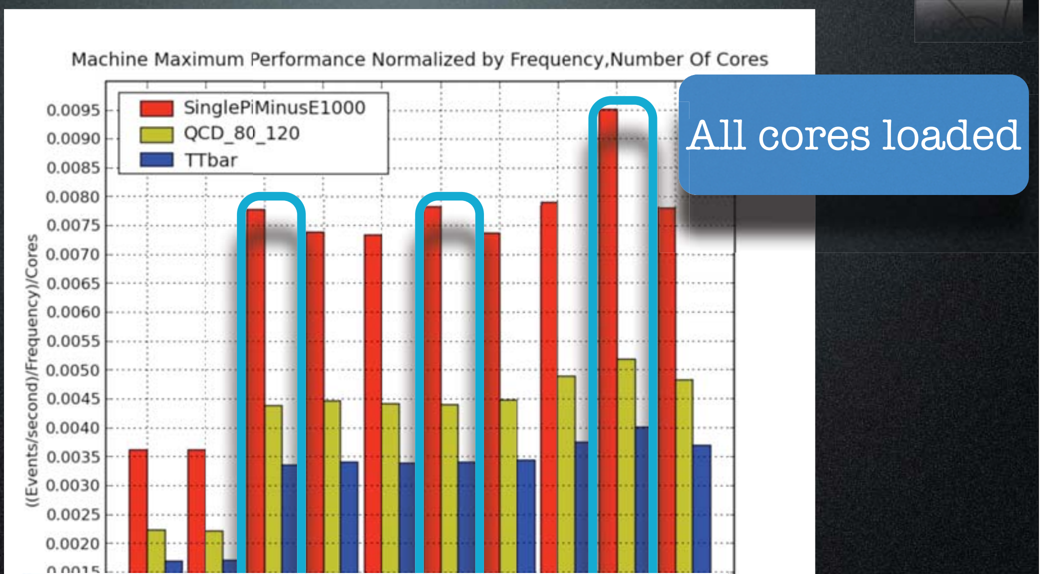
Comparing GEN+SIM



GEN+SIM normalized by frequency

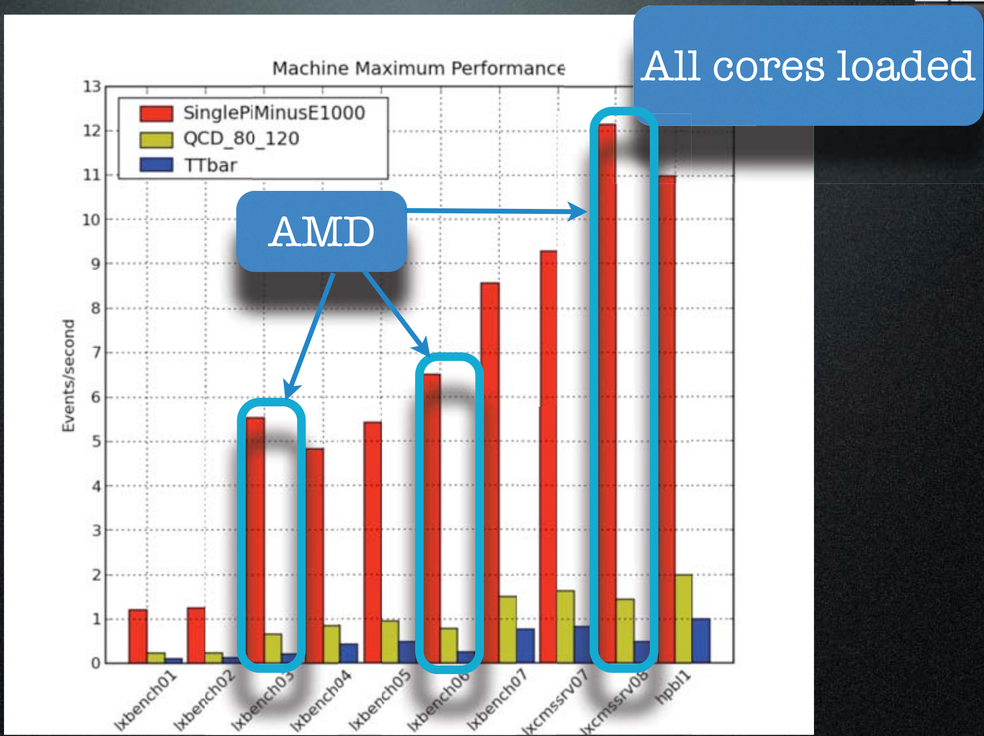


GEN+SIM normalized by frequency and # cores

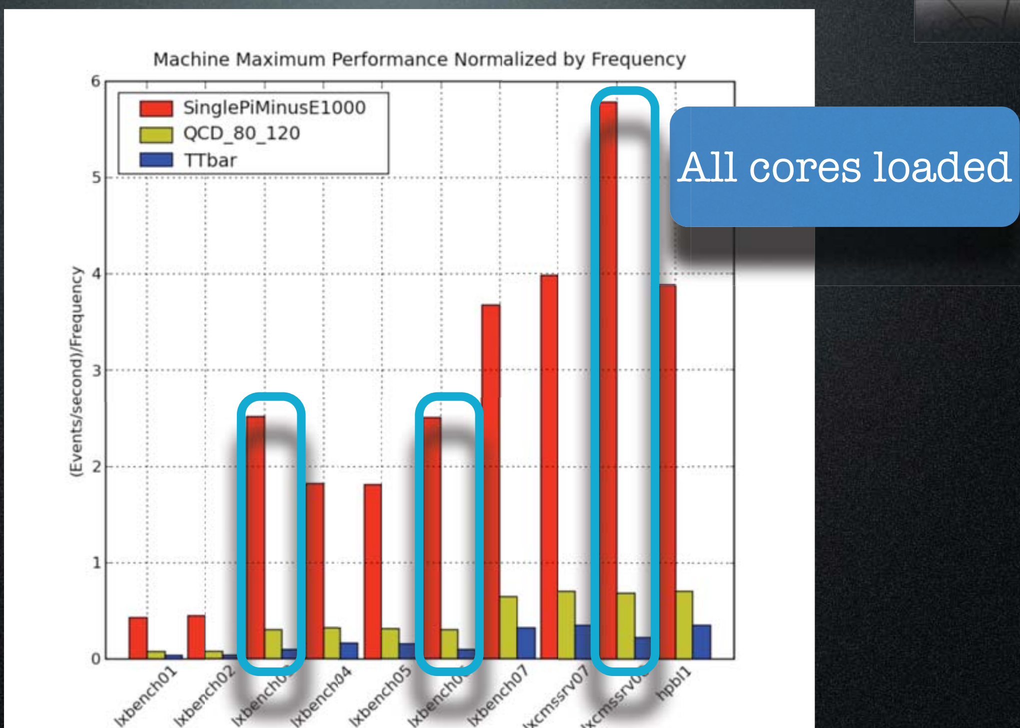


	lxbench01	lxbench02	lxbench03	lxbench04	lxbench05	lxbench06	lxbench07	lxcmssrv07	lxcmssrv08	hpbl1
Number of cores	2	2	4	4	4	4	8	8	8	8
Frequency (GHz)	2.8	2.8	2.2	2.66	3.0	2.6	2.33	2.33	2.1	2.83
Cache (could be L2/L3) (MB)	1	2	2	4	4	2	8	12	2	12
Memory (GB)	2	4	2	8	8	8	16	16	16	16
Processor	Nocona	Irvingdale	Opteron 275	Woodcrest	Woodcrest	Opteron 2218 Rev.F	Clovertown	Xeon HarperTown E5410	Opteron Barcelona 2352	Xeon HarperTown E5440
Vendor	Intel	Intel	AMD	Intel	Intel	AMD	Intel	Intel	AMD	Intel

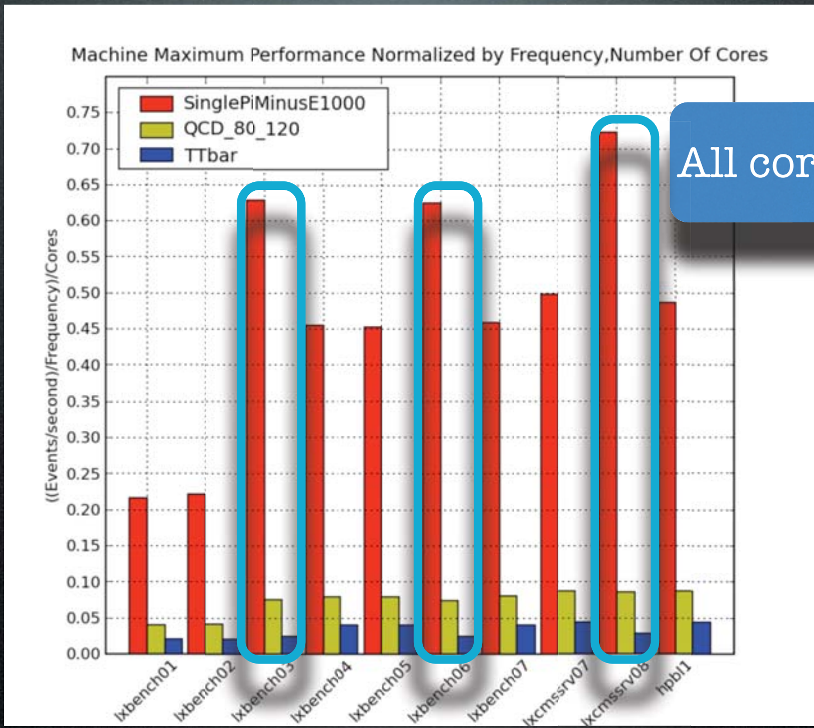
Comparing RECO



RECO normalized by frequency

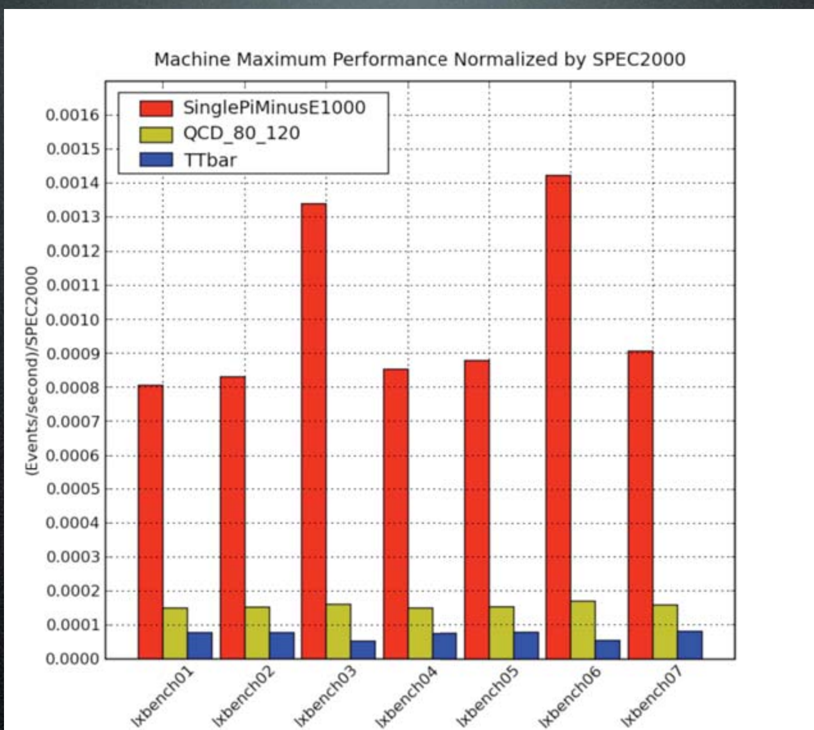


RECO normalized by frequency and # cores

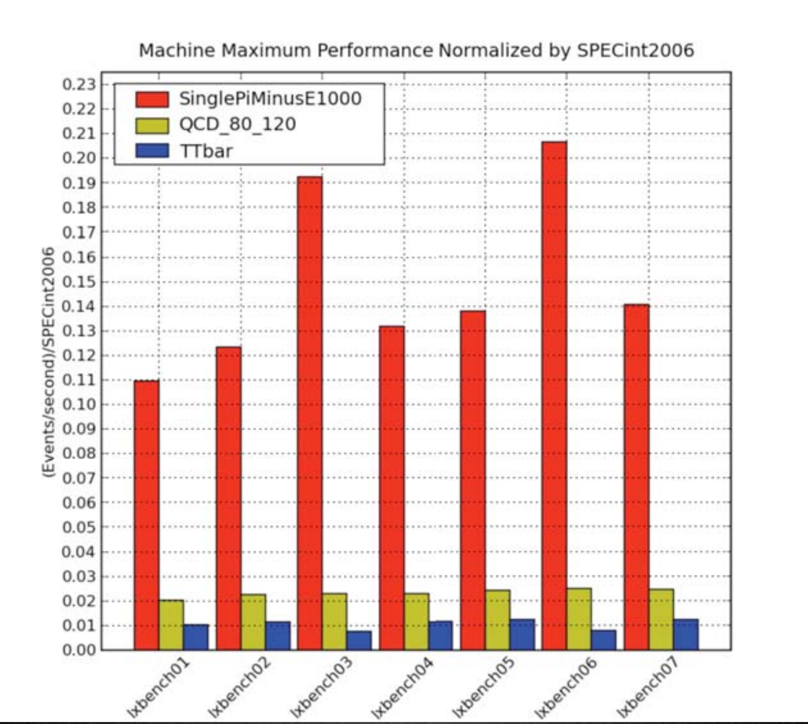


All cores loaded

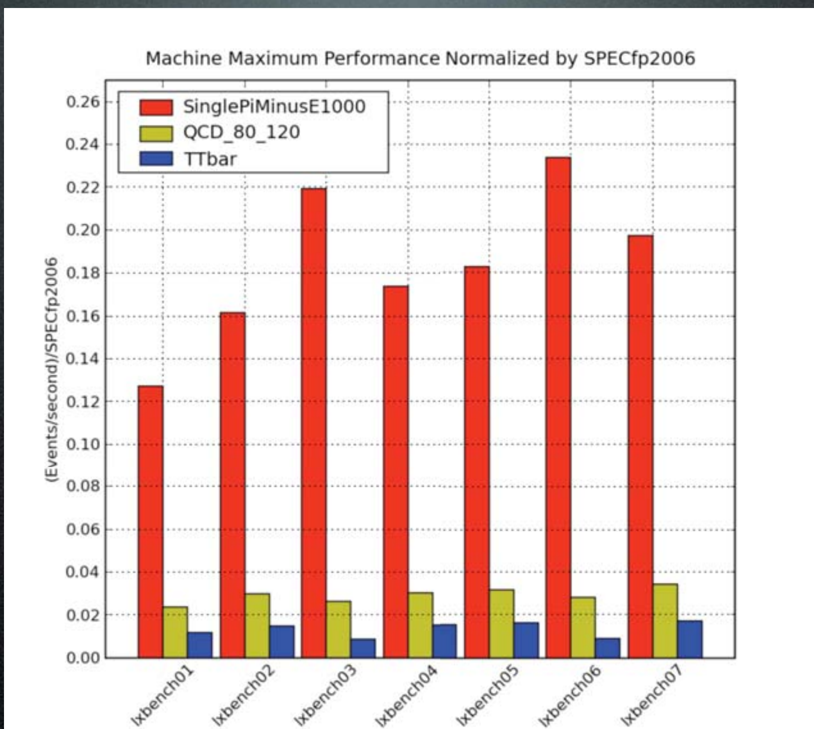
RECO vs SPEC2000



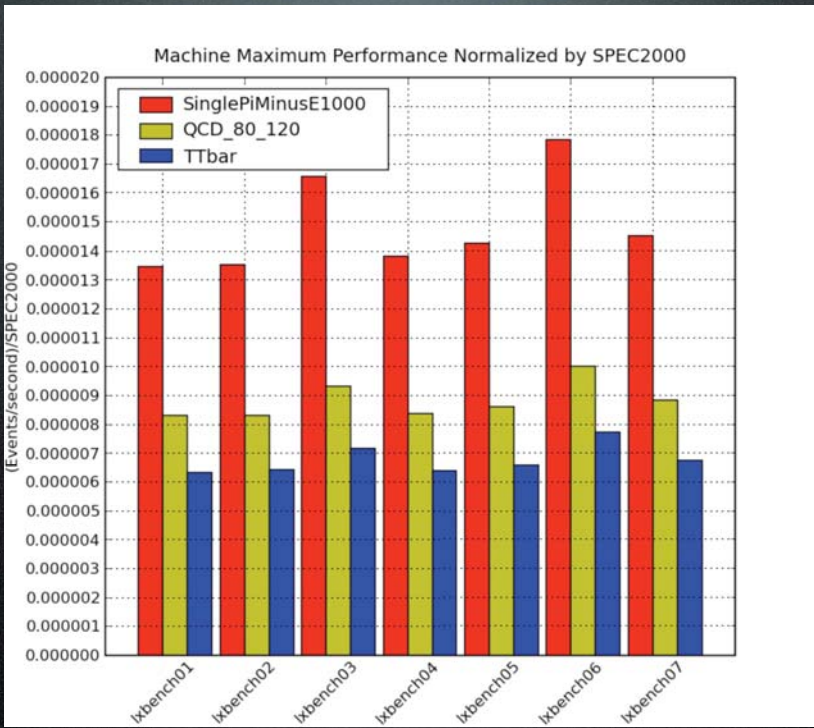
RECO vs SPECint2006



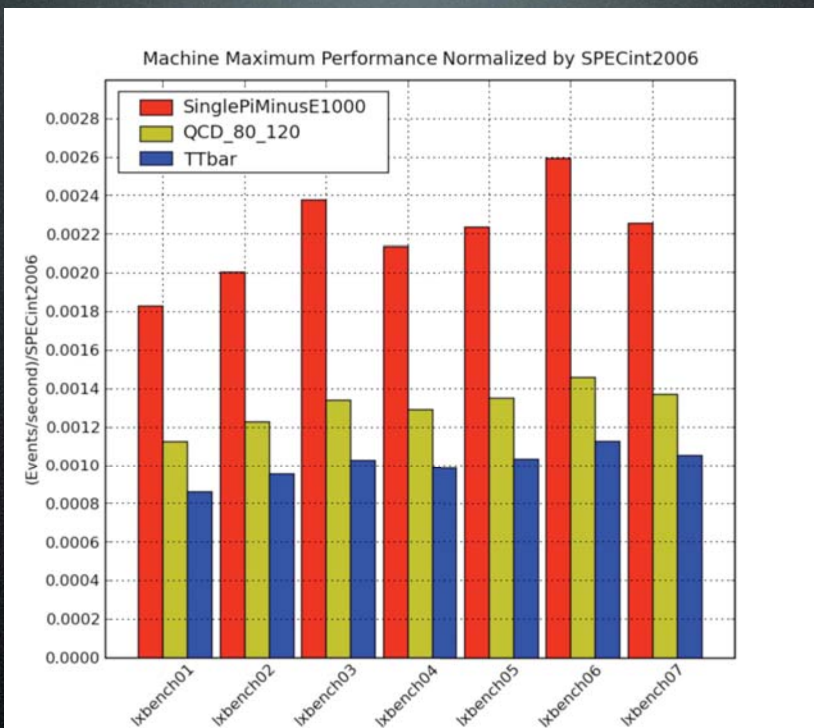
RECO vs SPECfp2006



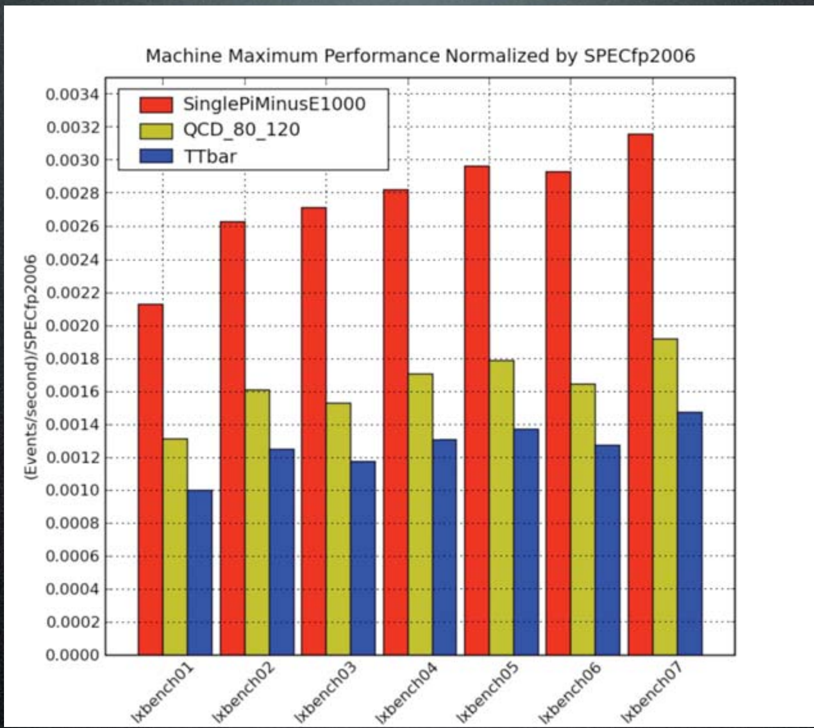
SIM vs SPEC2000



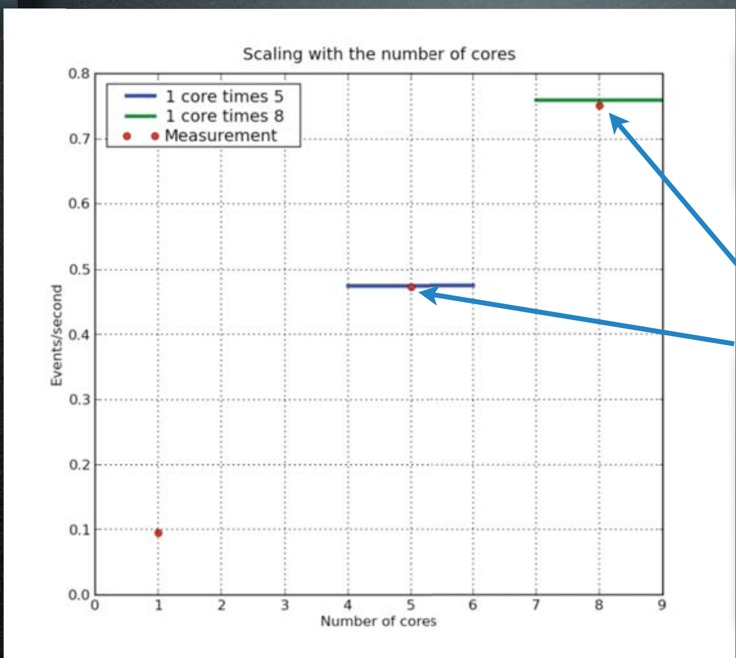
SIM vs SPECint2006



SIM vs SPECfp2006



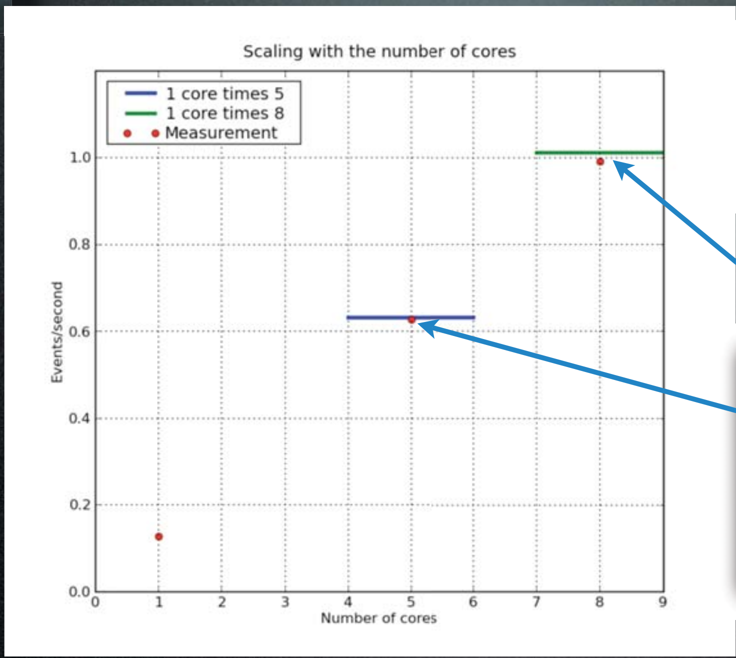
Scaling with multicores



lxbench07 TTbar RECO

Excellent scaling to the last core (cmsScimark2)

Scaling with multicores



hpbl1
TTbar RECO

Excellent scaling to the last core (cmsScimark2)

CMSSW Benchmarking Results

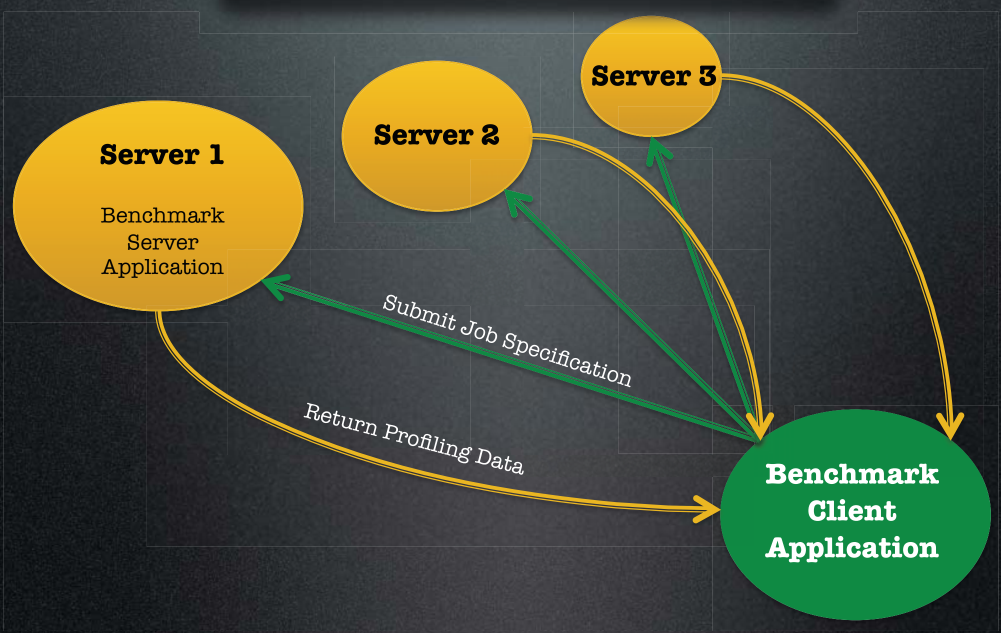
- Observed a different behavior in AMD vs. Intel machines for complex vs. simple events at the RECO step
- Compared CMSSW applications with SPEC benchmarks: differences due to architecture/type of event are larger than the differences between different SPEC benchmarks
- The CMSSW application scales nicely with the current multicore architectures
- A number of open issues from this first experience:
  - Statistical treatment of the data (number of events used, reproducibility, significance of the measurements)
  - Interpretation of the results ("one score" benchmark, weighting of several scores, picking representative candle(s), scores)
  - Graphical/data analysis
- The results of this work, done earlier this year, has inspired the development of a CMSSW benchmark utility that would address these issues and provide the necessary functionality to be used by Tier centers in assessing the CPU performance of machines

CMSSW benchmarking tools

- The idea is to have included in the CMSSW release a “suite” of benchmarking tools
- The basic functionality would be to run a special command of the Performance Suite, then harvest the log files for the CPU timing information
- The command above would return a score (maybe with its composition, in terms of multiple candles, or multiple processing steps, with relative weights, so that a Tier1 vs Tier2 could decide which score is most relevant for their use scenario)
- Since this kind of benchmarking usually involves more than one machine (since one is interested in comparing them), we thought of implementing server-client communication via XML-RPC in Python.
- Finally the data from multiple machines would have to be harvested from the logfiles, analyzed and reported in plots and scores

Client/Server Benchmarking

Using XML-RPC Transport



CMSSW benchmarking tool

- Then all any user would have to do, is to follow the APT based installation procedure for CMSSW on each machine intended to be benchmarked.
- On each machine launch an XMLRPC server
- On one machine launch the XMLRPC client, that ships the commands for the Performance Suite to the servers
- Wait for the Suite to finish on each machine (could do multicore running, or multiple runs on the same core(s)) and to report all results in term of a pickled file that contains dictionary data structure
- On this data structure run by default a basic analysis that would produce one score, a table with all results and the relevant comparison plots.
- Up to now the client/server functionality is under testing, the data analysis and score composition requires still some work

Conclusions

- The Performance Suite is an integral part of the CMSSW Release Validation process, providing developers with in-depth performance results and regression information within a 24 hrs cycle
- The Performance Suite can handle a lot of profile data and summarize it making it available and usable by developers,
- Besides its default behavior, it can be tweaked for more statistics, changing tests, single candles, new versions of external tools, new profiling tools, it can also be used as the core of the CMSSW benchmarking utility
- The framework is ready to handle the first collision data . A few improvements are in the works and the performance suite is used in testing and guiding optimization
- A command-line and XML-RPC based client/server CMSSW architecture benchmarking suite is being implemented and will soon ship with the release

Back-up

The CMS Experiment

June 2008

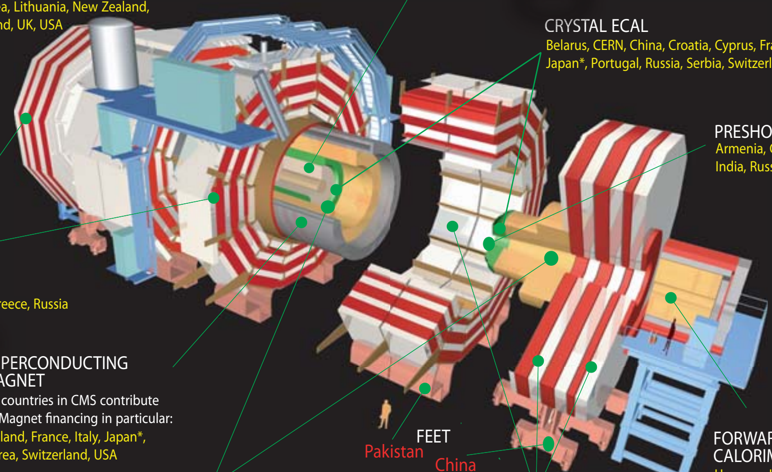
38 Countries, 183 Institutes, 3000 scientists and engineers (including 400 students)

TRIGGER, DATA ACQUISITION
& OFFLINE COMPUTING
Austria, Brazil, CERN, Finland, France, Greece, Hungary, Ireland, Italy, Korea, Lithuania, New Zealand, Poland, Portugal, Switzerland, UK, USA

TRACKER
Austria, Belgium, CERN, Finland, France, Germany, Italy, Japan*, Mexico, New Zealand, Switzerland, UK, USA

CRYSTAL ECAL
Belarus, CERN, China, Croatia, Cyprus, France, Italy, Japan*, Portugal, Russia, Serbia, Switzerland, UK, USA

PRESHOWER
Armenia, CERN, Greece, India, Russia, Taiwan



RETURN YOKE
Barrel: Estonia, Germany, Greece, Russia
Endcap: Japan*, USA

SUPERCONDUCTING MAGNET
All countries in CMS contribute to Magnet financing in particular: Finland, France, Italy, Japan*, Korea, Switzerland, USA

FEET
Pakistan
China

FORWARD CALORIMETER
Hungary, Iran, Russia, Turkey, USA

HCAL
Barrel: Bulgaria, India, Spain*, USA
Endcap: Belarus, Bulgaria, Georgia, Russia, Ukraine, Uzbekistan
HO: India

MUON CHAMBERS
Barrel: Austria, Bulgaria, CERN, China, Germany, Hungary, Italy, Spain
Endcap: Belarus, Bulgaria, China, Colombia, Korea, Pakistan, Russia, USA

Total weight	: 12500 T
Overall diameter	: 15.0 m
Overall length	: 21.5 m
Magnetic field	: 4 Tesla

* Only through industrial contracts

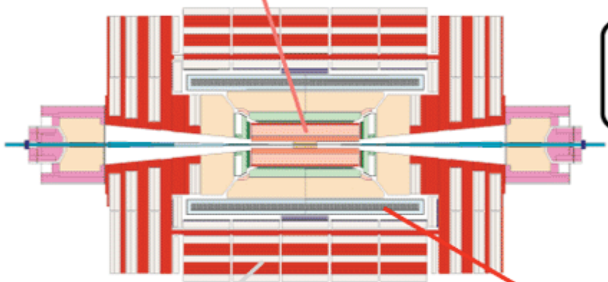
Germany in CMS

6 Institutes
216 Collaborators
As of September 2008

TRACKER
Detector, electronics, mechanics, support and monitoring
IEH, IEKP, PI3, PI3

HIGH LEVEL TRIGGER
DESY

OFFLINE COMPUTING
IEH, IEKP, PI1, PI2, PI3



MUON Barrel Driftubes
Construction and assembly, installation, slow control system, gas and cooling system
PI2

MAGNET
Procurement
IEH, IEKP, PI1, PI2, PI3

KEY: IEH - Institut fuer Experimentalphysik, Hamburg; IEKP - Institut fur Experimentelle Kernphysik, Karlsruhe;
PI1 - I. Physikalisches Institut, Aachen; PI2 - III. Physikalisches Insitut A, Aachen; PI3 - III. Physikalisches Institut B, Aachen;
DESY - Deutsches Elektronen-Synchrotron, Hamburg

3 slides each (CPU, Size, Memory), maybe 3,2,4 with an example of use of the tools to investigate the matter/ improve (tomorrow submit 2_1_4 to compare with 2_1_10 for SIM and RECO
CPU:
1-TimevsEvt*+histo for MinBias SIM? (mention possibility to skip first event?) with and without regression
In the same TimeReport breakdown by module
(Tabella con tutte le candele)
IgProfPerf con Regression
2-Size Tabella con tutte le sizes to see progressive increase, EdmSize with Regression
3-Memory
SimpleMemcheck:
(Pile-up?), example of bumps and coming down, comparison between successive steps, or Pile-up
IgProfMem con Regression

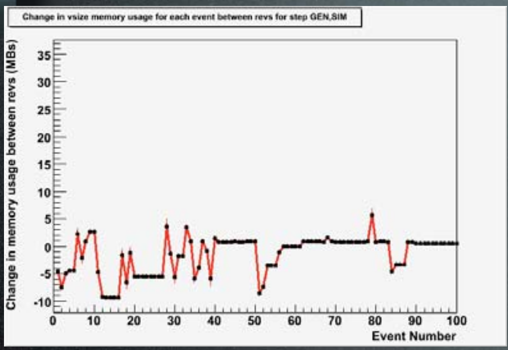
Points to make:
1-Automated procedure to measure and monitor performance
2-Part of Release Validation QA
3-Quick response time/enough detailed information
4-Wide coverage of code, candles, processing steps
5-Robust (based on RelVal samples), options added
6-USed also for other tests
7-Re-used for benchmarking purposes
8-Can include robustness and reproducibility tests

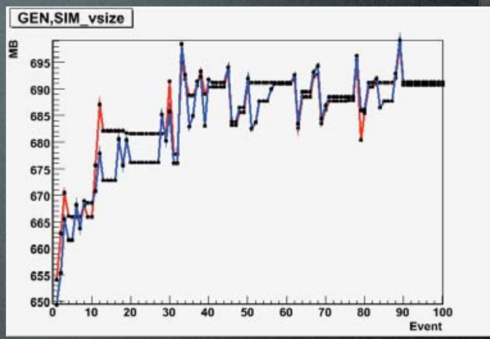
9-All above... but we need to present

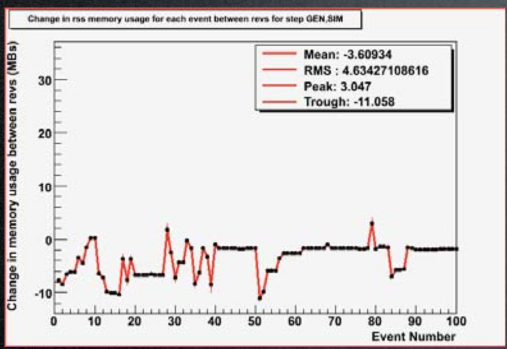
Add Graphics with the structure of the Suite itself? OR Graphic with the issues of power saving paranoia (benchmark before and after on the core), stuff on the other cores

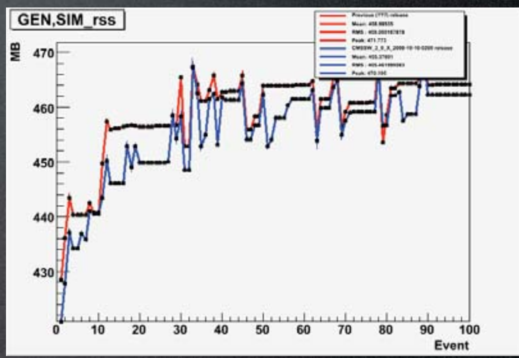
Key point in Results:
1-CPU Time: a table and a couple of plots
2-File Size a regression plot of EdmEventSize or ls table
3-Memory a plot of SimpleMemoryCheck/Total memory estimate from MemTotal (?)

Memory Performance



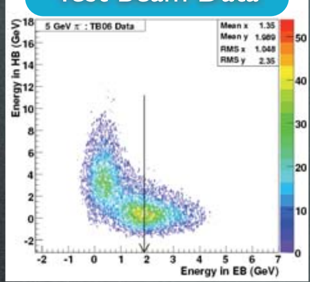




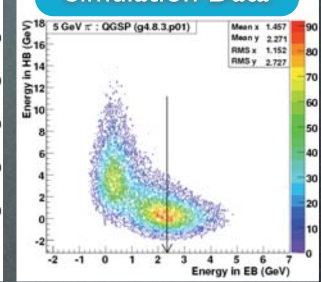


Simulation Test Beam Validation

Test Beam Data

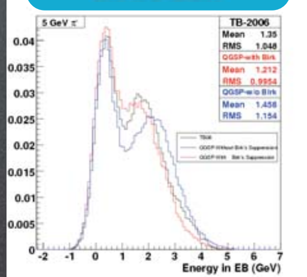


Simulation Data



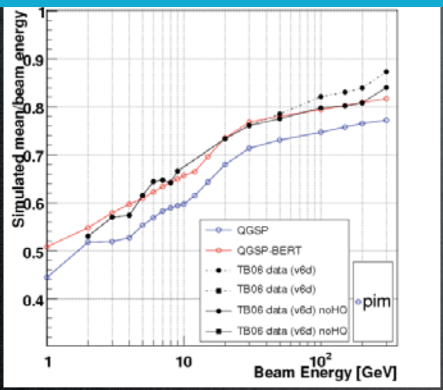
Birk's Law

Scintillator Saturation Effects

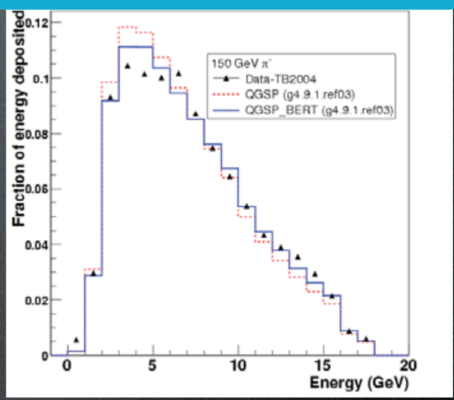


Simulation Test Beam Validation

Mean Energy Response
ECAL+HCAL with pion beam



Longitudinal Shower Profile
HCAL



Simulation Test Beam Validation

GEANT4 Physics Lists	CPU Time (%)		Event Size (%)	
	MinBias	TTbar	MinBias	TTbar
QGSP_EMV	100	100	100	100
QGSP	116	120	101	103
QGSP_BERT_EMV	141	146	152	177
QGSP_BERT	158	169	152	172

A little bit of history

- Until the last release cycle (21X) CMSSW used a special configuration language to configure the (one and only) cmsRun executable
- The full transition to Python happened with 210 and came with a major improvement in maintainability: cmsDriver.py
- This script is a command-line utility that prepares a full python configuration file, based on a few command-line options, and launches cmsRun on it.
- This is highly configurable and covers most use cases

Details on Python tools

- Graphics to show the cmsDriver, cmsRelvalreport, cmsPerfSuite, cmsRelRegress, cmsPerfRegress, cmsRelvalreportInput ...
- Examples of uses of one and the other (customize fragment and its effect in the log, parsers and plot drawing, regression and publishing on a webserver)
- Issues about the machines used to do the measurements (for CPU only, but also for memory in case of ununderstood crashes)
- Uses of the suite for other aims:
  - tests on external packages (G4 reproducibility, its performance, robustness),
  - use as a machine benchmarking building block (moving to the second part of the talk)

Fast!