

An LHC Computing Sampler (And thoughts about BaBar/SuperB)

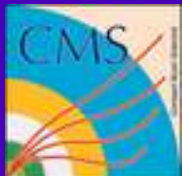
Peter Elmer
Princeton University
SuperB Workshop, La Biodola, Isola d'Elba
1 June, 2008



Overview



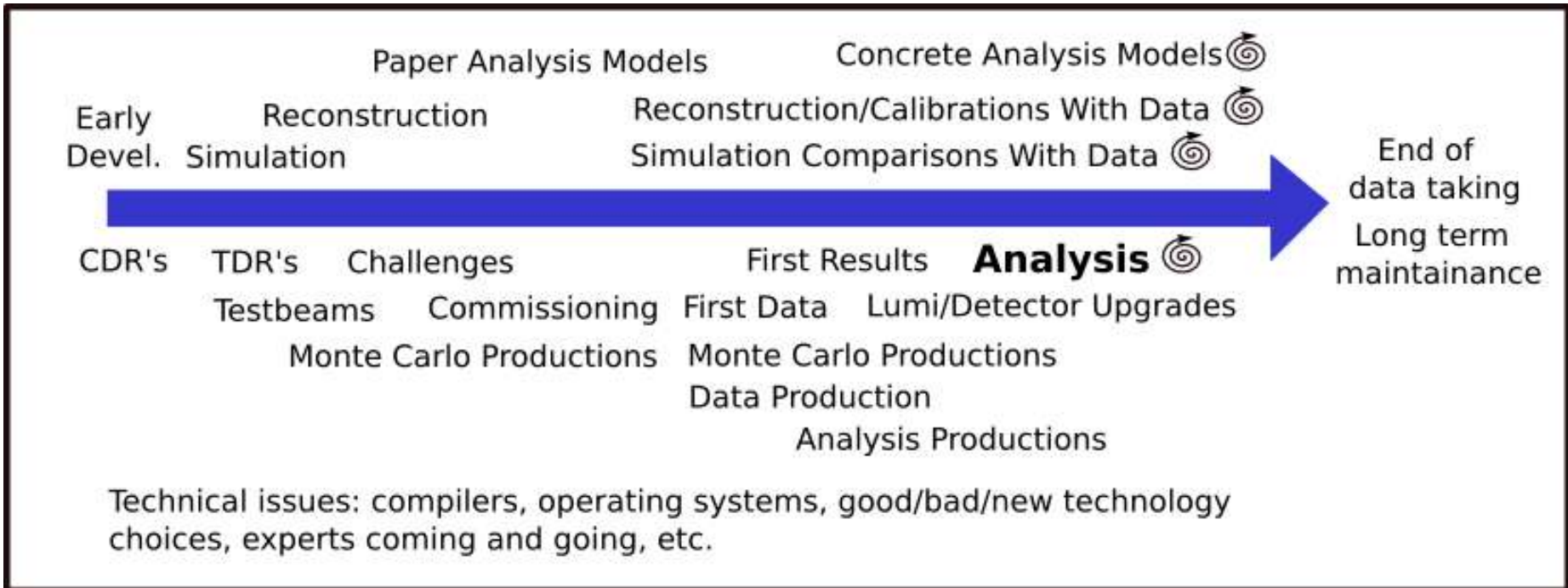
- I'm not really going to give a complete summary of all-things-LHC in this presentation, and not even all-things-CMS, but instead intend to cover:
 - Some slides about the software life cycle (BaBar and CMS)
 - Some notes about various software choices we made in CMS, and where they differ from BaBar
 - I'm not going to cover “grid” things at all
 - Some things I find interesting (and which we are pursuing to varying degrees for CMS) as R&D-like projects



HEP Software Life Cycle



- Last summer, as a follow-on to the CMS changes, I did some basic studies with Liz Sexton-Kennedy (CDF), Chris Jones and Dan Riley (CLEO) to compare BaBar/CDF Run 2/CLEO III/c with CMS.





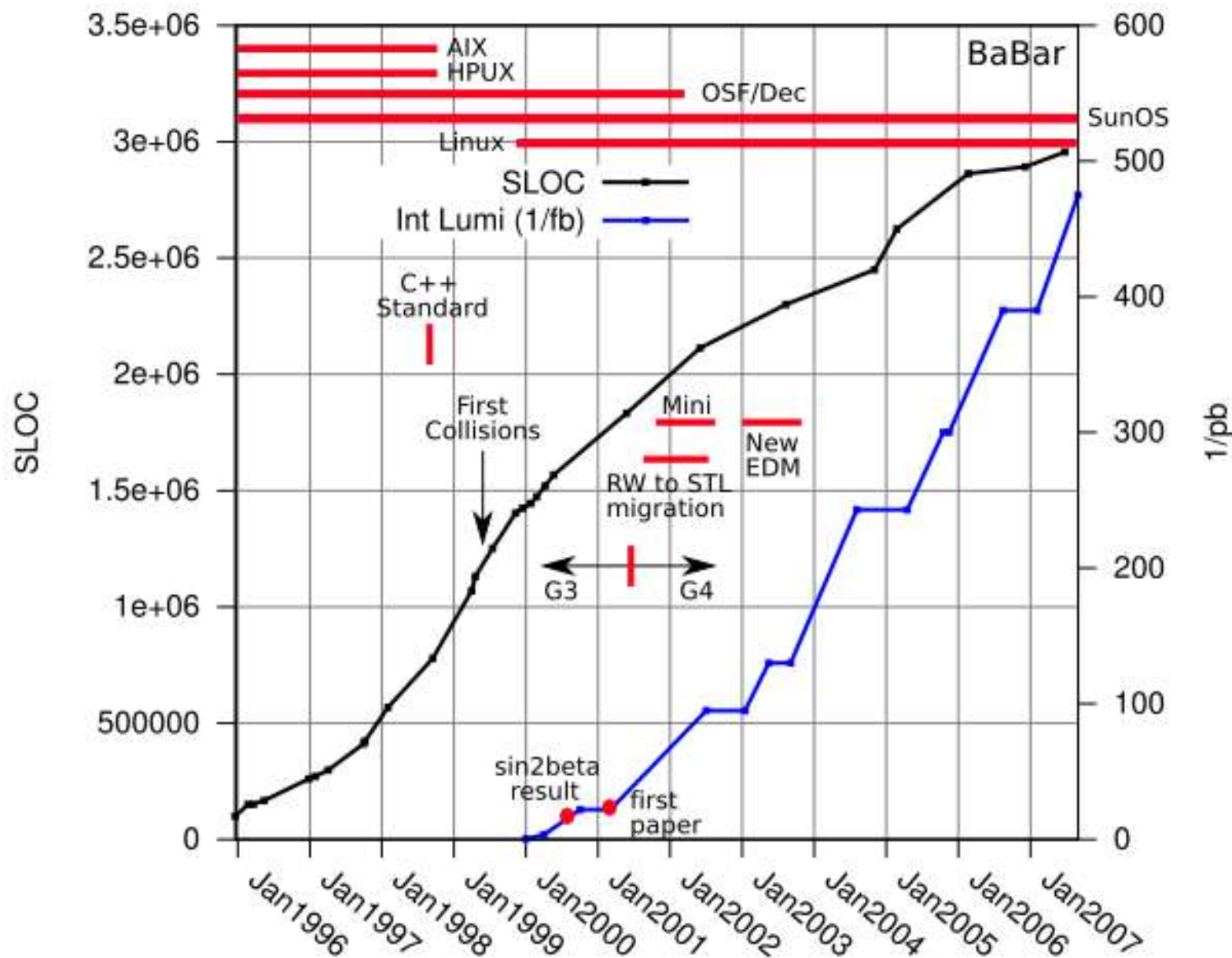
HEP Software Life Cycle

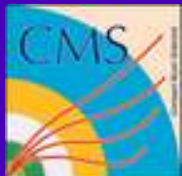


- Mostly it was educational to look back through what happened in those experiments, but we also put together a presentation of some of the quantitative aspects of that for CHEP2007.
- The full presentation, mostly showing metrics about the software development, can be found at:
<http://indico.cern.ch/contributionDisplay.py?contribId=351&sessionId=28&confId=3580>
- We basically looked at the evolution of the code base in terms of size (source lines of code = SLOC), the number of people working on it and also (approximately) how it was distributed in terms of code type/language. We looked at releases, not CVS.
- I'll show the BaBar/CMS statistics in the following slides...

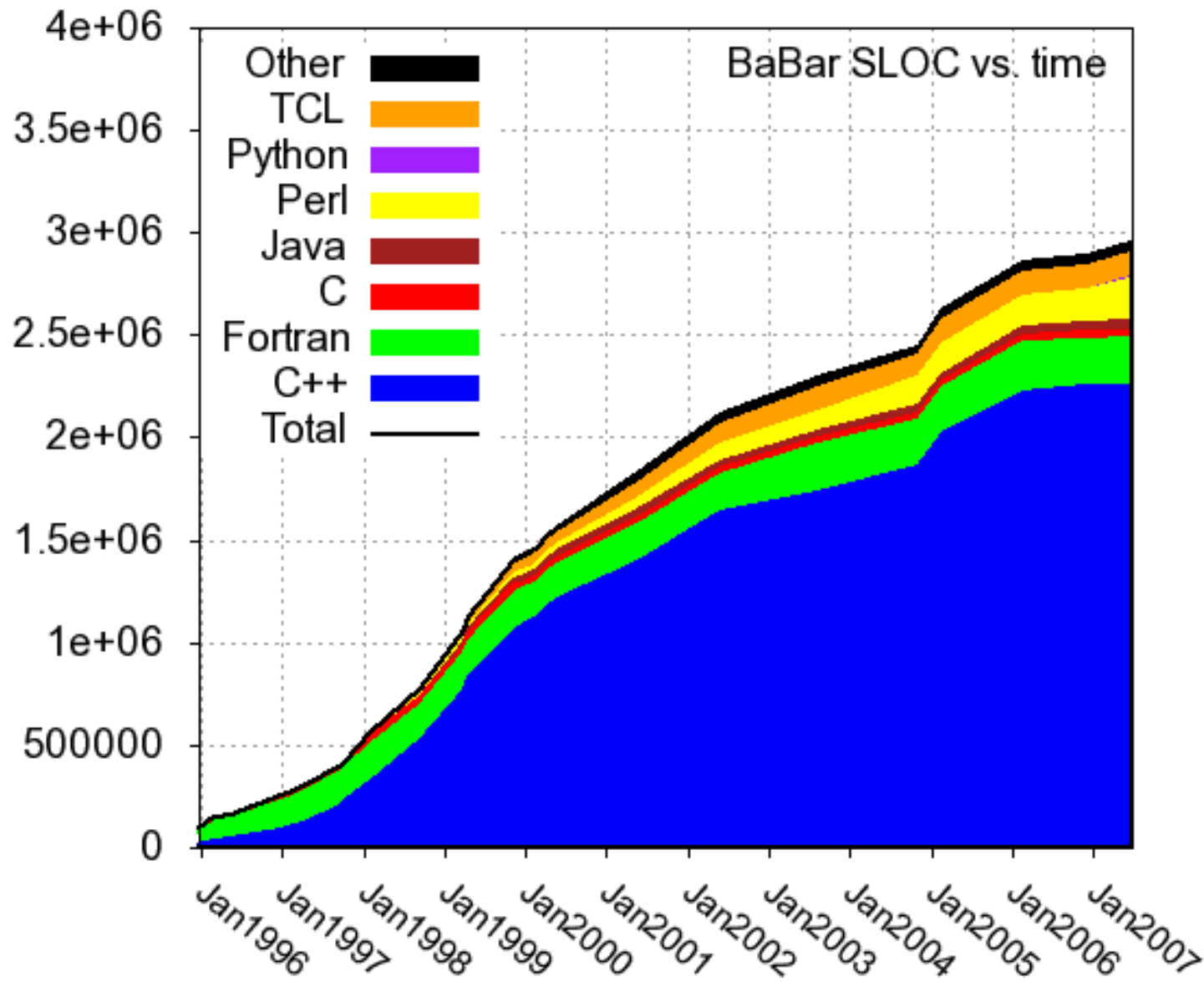


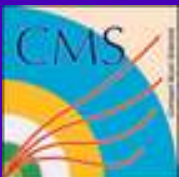
BaBar SLOC and Luminosity



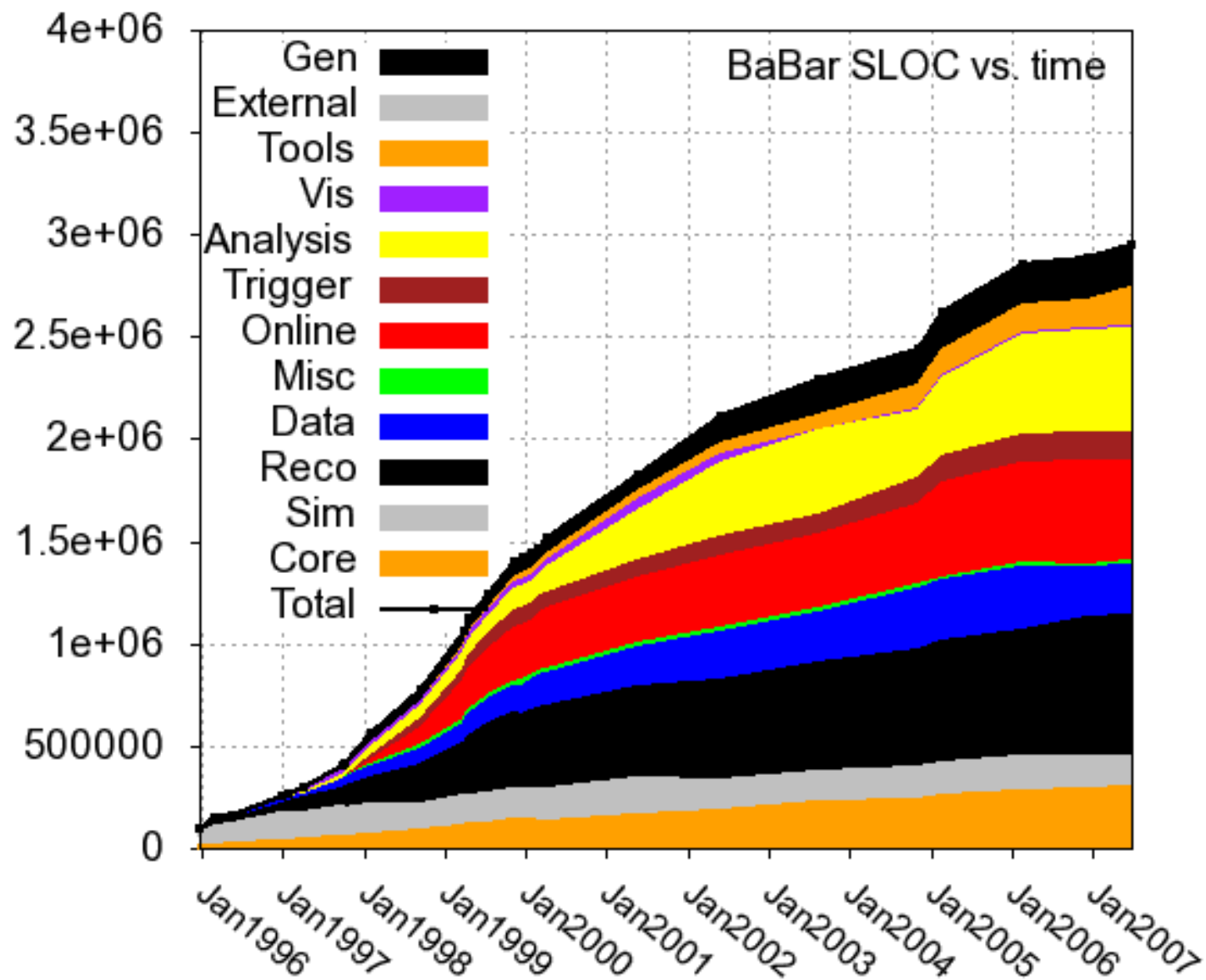


BaBar SLOC and language



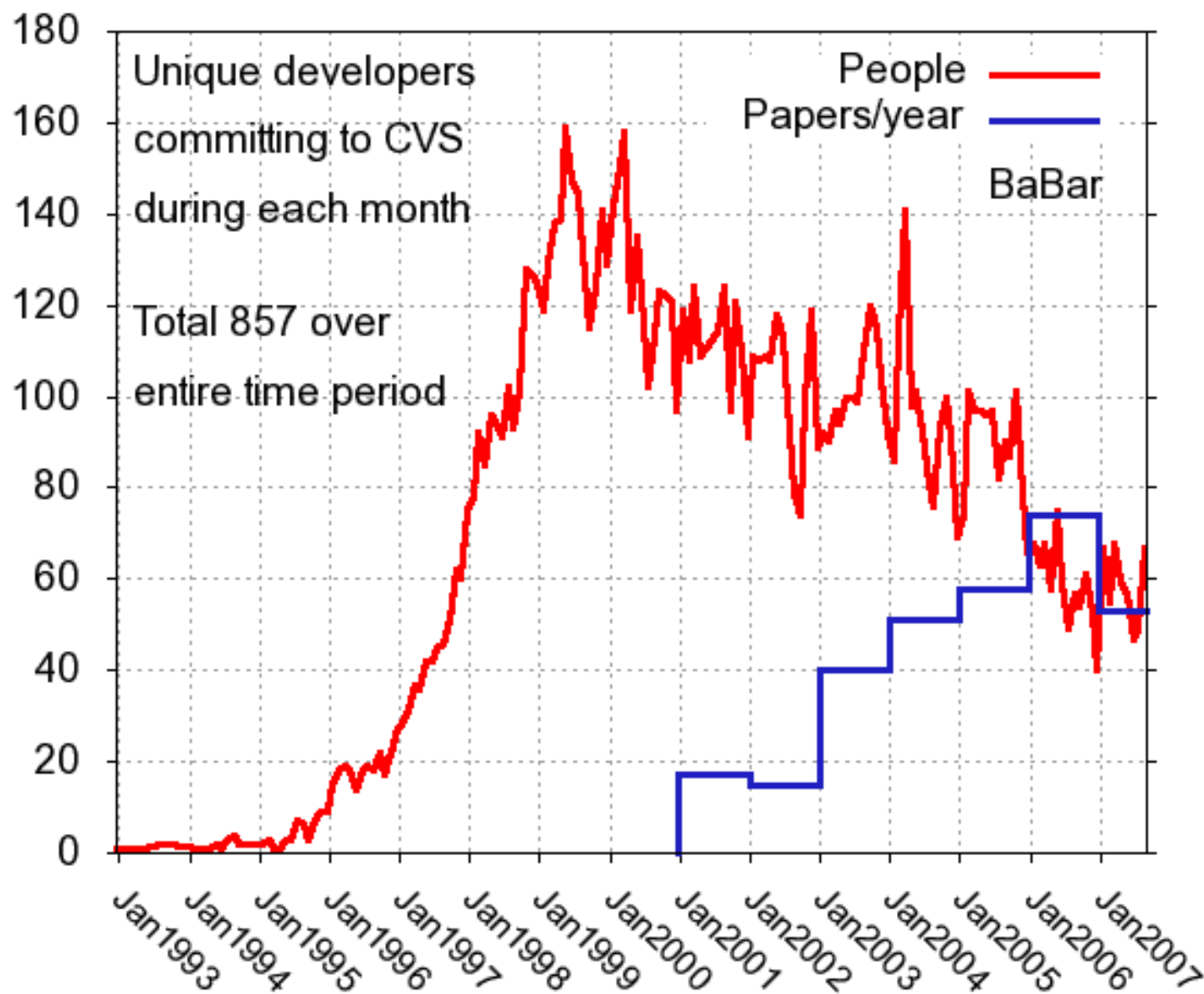


BaBar SLAC and code type





Developers committing/month

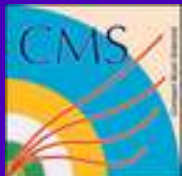




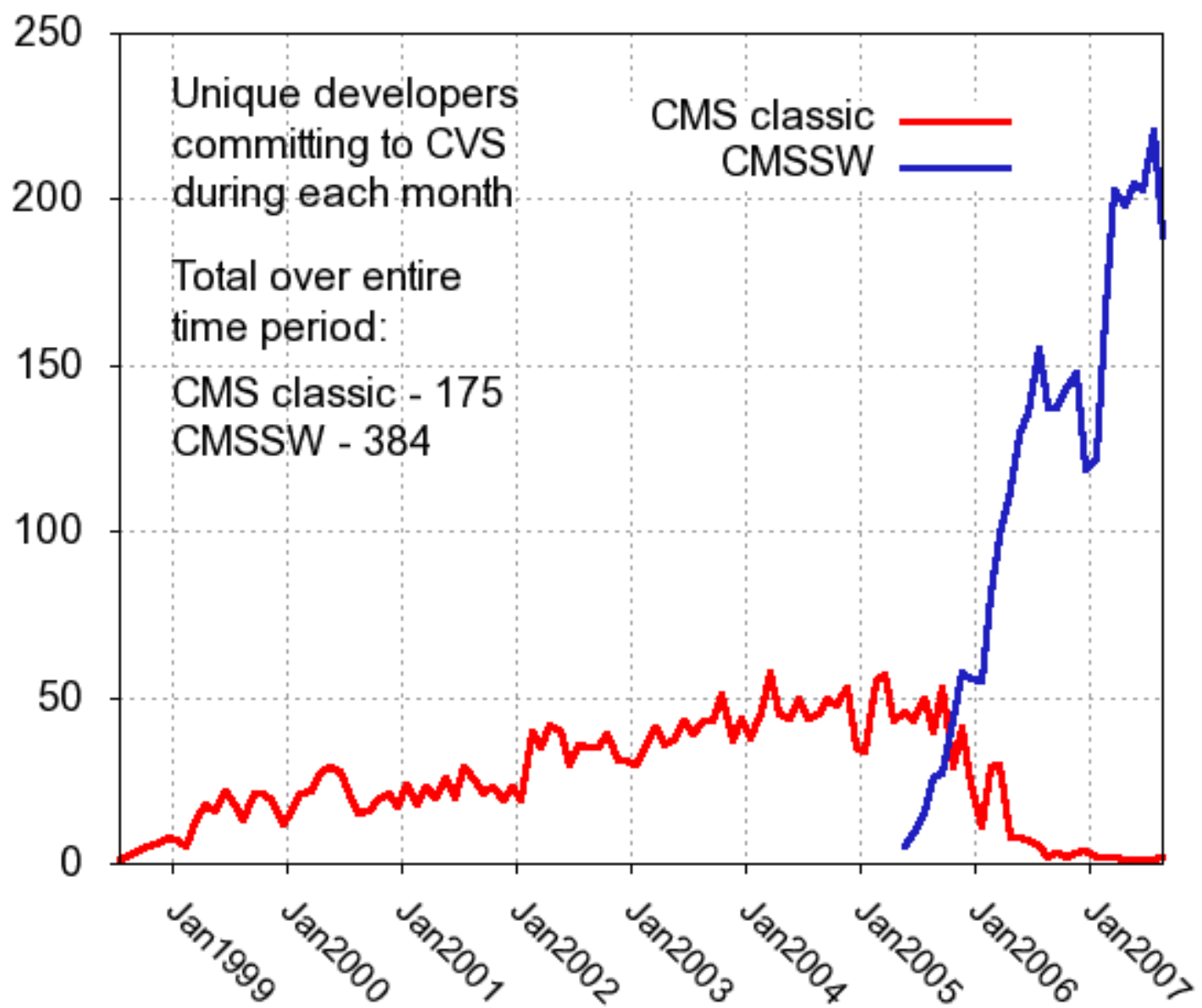
CMS



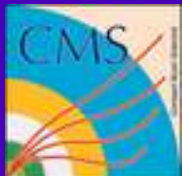
- CMS (like the other LHC experiments) has a software/computing effort which dates back to the same time period in which BaBar/CDF Run 2/CLEO III/c were starting.
- Following various problems that became apparent in a CMS (mock) data challenge in 2004, however, it was decided to do a rather large “restart” for the software, framework, event-data-model and workflow/data management. This effort ramped up during 2005 and is by now the standard software/computing environment for CMS.
- Nearly all of what I'll say here about CMS comes from the later period, of course.



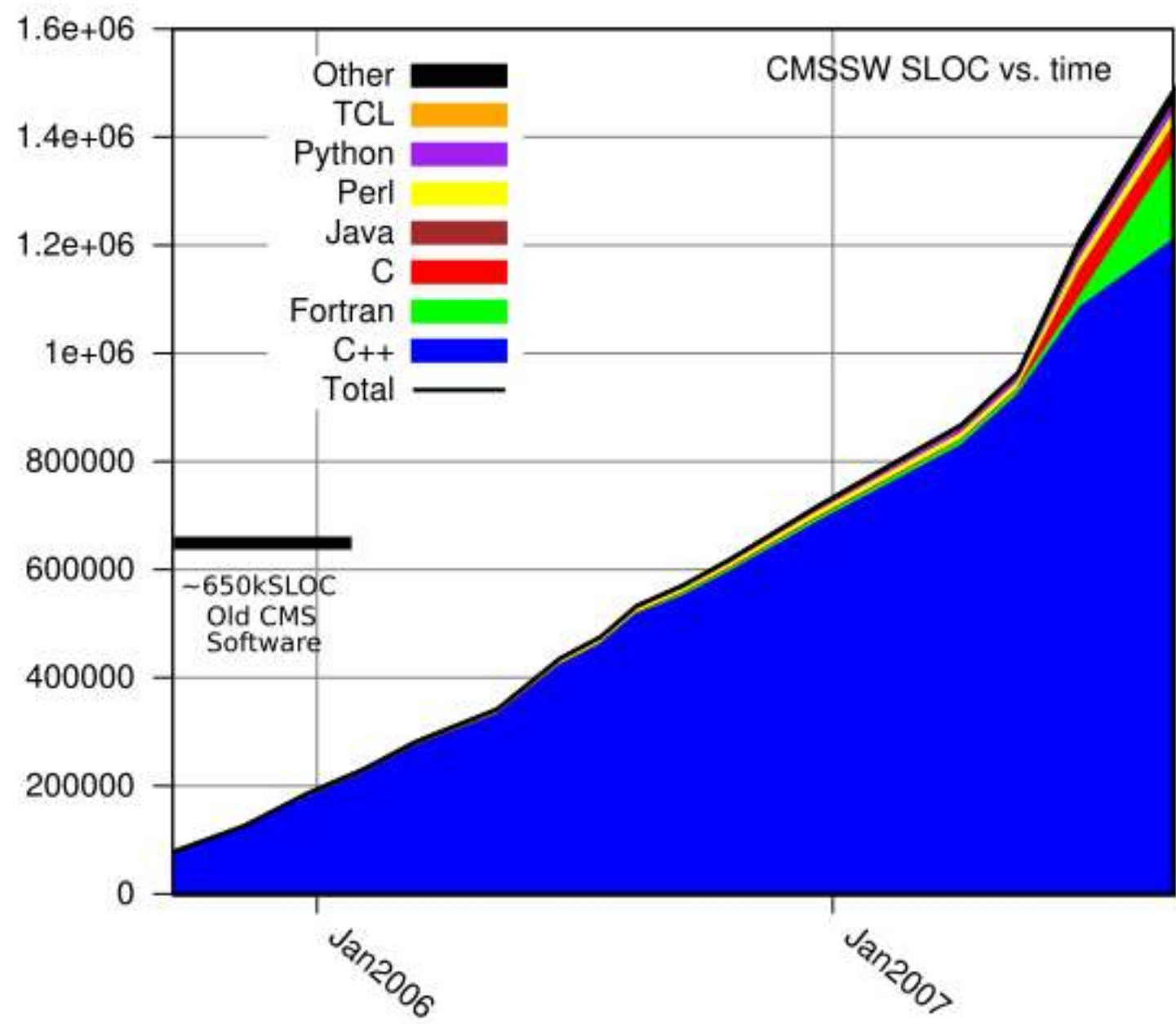
CMS developers/month



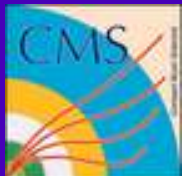
These are statistics through Sept. 2007, more recently we have hit ~240 people committing per month



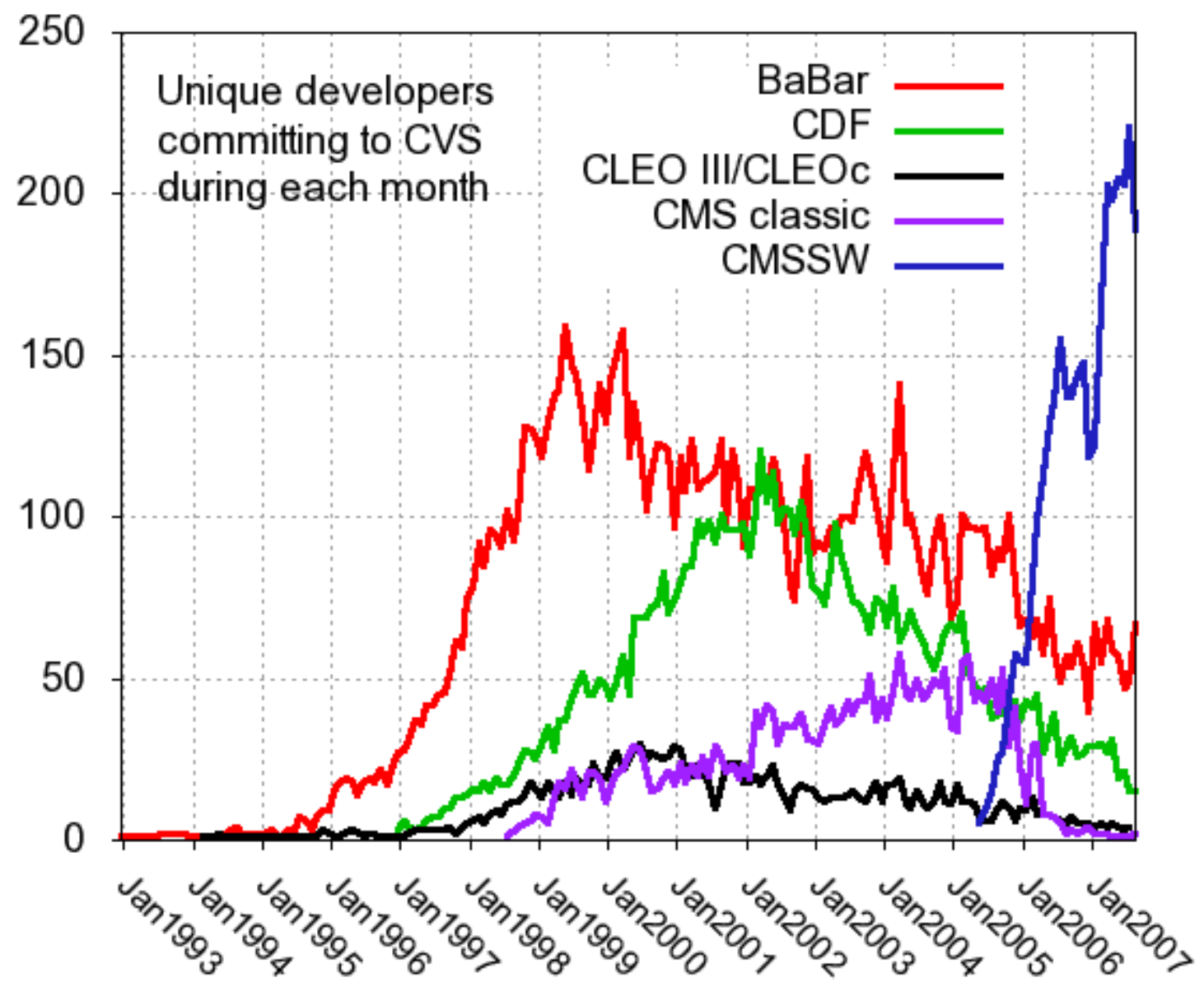
CMS SLOC vs time

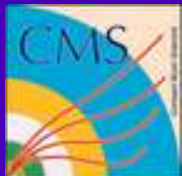


Statistics through Sept, 2007.
More recently we have hit ~2MSLOC



Aside: effort on all expt's vs time





Communication



HyperNews

Forum: Data Quality Monitoring Development Member: elmer (logout)

	Forums by Category	Recent Postings	Member Info	Overview
	Forums by Time Order	Search in Forums	Members List	Contact Admin
	Request a New Forum	Subscribe to Forums	New Member	

Discussion about the development of the Data Quality Monitoring infrastructure (including feature requests, bugs, etc)

The email gateway for this forum is: hn-cms-dqmDevel@cern.ch

Show subscribers

Inline Depth: 1 All Outline Depth: 1 2 All Add message: +

- 48 [Threading problems with clients \(again\)](#) (Christos Leonidopoulos - May 04, 23:41) NEW
 - 1 [Re: Threading problems with clients \(again\)](#) (Suchandra Dutta - May 04, 23:48) NEW
 - 1 [Re: Threading problems with clients \(again\)](#) (Christos Leonidopoulos - May 04, 23:53) NEW
 - 2 [Re: Threading problems with clients \(again\)](#) (Emilio Meschi - May 05, 10:48) NEW
 - 1 [Re: Threading problems with clients \(again\)](#) (Emilio Meschi - May 05, 10:50) NEW
 - ... 10 Message(s) NEW
 - 2 [Re: Threading problems with clients \(again\)](#) (Christos Leonidopoulos - May 05, 10:52) NEW
- 47 [Clients: monitoring cycles and modified contents](#) (Christos Leonidopoulos - Apr 29, 13:13)
- 46 [Re: Porting from ORCA to CMSSW](#) (Christos Leonidopoulos - Apr 28, 10:57)
 - Alert for package "DQM/SiStripCommon" (Christos Leonidopoulos - Apr 28, 09:57)
 - de" (Christos Leonidopoulos - Apr 28, 01:15)
 - ff-line mode" (Giuseppe Della Ricca - Apr 28, 15:11)
 - ff-line mode" (Christos Leonidopoulos - Apr 28, 23:18)
 - ff-line mode" (Giuseppe Della Ricca - Apr 29, 09:48)

TWiki

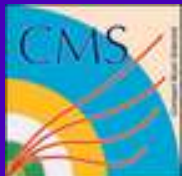
Phone conferences
 VRVS/EVO video
 conferencing
 IM/Chat




Software Development Model



- Even if we perhaps don't think about it we (HEP) have a particular (distributed) software development model
 - Widespread collaboration (i.e. distributed) participation in writing our own software.
 - Small number of full-time “computing professionals” covering some core functionalities (Framework, core Conditions model, event persistence, etc.)
- Participation in software development is the most “accessible” part for individual contributions to the collaboration and being able to use the software is necessary for analysis. Facilitating this is what the model does.
- I don't think there are many new things in CMS here, in fact over the past 3-4 years it was steered towards something that is a bit like BaBar.



“CMS Tag Collector”



CMS Tag Collector: Release Management Page

Peter Elmer (elmer)
Thu Jul 13 04:42:37 CEST 2006
Privilege Level: 3

CMS Tag Collector Links

[Release Management](#)

[Package Publication](#)

[Create New Release](#)

[Create New Package](#)

[Administration](#)

[Edit User Preferences](#)

[Logout](#)

[Doc](#)

CMS Tag Collector maintained by:
[David Lange](#)

Ported for CMS from [WebSRT](#).

The procedure for updating tags in the system is:

1. Create a Publication for each new tag (comparing the old system with the new: Publication = Checklist + Package Announcement): Go to the Publication page link in side bar [or make "Publication page" a link]. Complete the form, then click on "Submit". Return to the Release Management Page. You then can change package versions in particular releases.
2. Select the release(s), package(s), and your new tag(s) below. The Releases menu gives you the option of putting the tag into any Open release. Tags can be queued for Closed releases.
3. Click "Submit Release Changes" below.

NEWS

The pulldowns for the package versions in the table below are now limited to the 10 most recent versions. Limiting the number of versions speeds up the formatting of this page.
If you use "Select Releases", only the selection(s) in the box will be shown after you click "Update Page".

Select Release(s):
(Hold the Control Key to make more than 1 selection or to add selections to the current list.)

nightly

CMSSW_0_9_0_pre1

CMSSW_0_8_0

CMSSW_0_8_0_pre4

CMSSW_0_8_0_pre3

CMSSW_0_8_0_pre2

CMSSW_0_8_0_pre1

CMSSW_0_7_2

CMSSW_0_7_1

CMSSW_0_7_0

Filter By Package Administrator

Filter By Package Filter

Enter Package filter:

Choose a Package Administrator::

The above parameters can be permanently changed by going to: [Edit User Preferences](#)

Usual difficulties in managing contributions from many distributed collaborators
 Twice daily integration builds, for multiple major release series and arches
 CVS (we unfortunately didn't make switch to subversion), SCRAM (like SRT)



CMS “external” software



- ignominy, iguana, rulechecker, boost, bz2lib, castor, clhep, cmake, coin, cppunit, curl, db4, dcap, doxygen, elementtree, expat, frontier_client, gcc, gccxml, gdbm, geant4, gpt, graphviz, gsl, gsoap, hepmc, heppdt, hippodraw, iodbc, java-jdk, libjpg, libpng, libtiff, libungif, meschach, mimetic, mysql, mysqlpp, openssl, oracle, oval, p5-dbd-oracle, p5-dbi, p5-libwww-perl, p5-template-toolkit, p5-uri, p5-xml-parser, pcre, python, qmtest, qt, qutexmlrpc, rfio, sigcpp, simage, soqt, sqlite, tkonlinesw, uuid, valgrind, xdaq, xerces-c, zlib, xrootd, SCRAMV1, coral, lcgaa, pool, root, seal, plus an ever increasing number of generators



Build/distribution system



- Managing all of these “externals” is quite difficult. The model in which there is an expert that knows how to build and install each package breaks down immediately. (This was true even in BaBar, where there is/was an expert for ROOT, Geant4, ACE/Tao, etc.)
- In CMS we have introduced a system in which the build recipes for all externals, plus the release itself, are maintained as rpm spec files. All special config options, arch-specific if's, patches, etc. are thus maintained (in CVS) so that at least the mechanical part of building them is reproducible. A small set of scripts is used to manage a set of spec files as a “configuration”.
- In addition both the externals and CMS software releases are packaged and distributed as rpms (installed in a non-root/non-system rpm DB!).
- apt is used to simplify the package management for the “user”. An 'apt-get' of a release will also automatically install the correct externals.



Framework



- Many of the concepts in the new Framework are similar to those in the BaBar Framework (also used by CDF)
- Some differences:
 - No need for AppUserBuild and relinking – modules are loaded via a plugin mechanism
 - Currently using a custom configuration language, but moving to python
 - EventSetup a la CLEO for managing intervals of validity for non-event data
 - “Service” system for various Framework decorations (e.g. equivalent of AppActions), also done via plugin mechanism



Framework



- Exceptions are used much more widely and propagated more cleanly to a “framework job report” for subsequent handling by the “workflow system” (i.e. MC production system, user analysis tools, etc.) `auto_ptr` is used fairly widely.
- Parameter defaults are handled outside the C++ code
- Most other things (modules, input/output, paths, etc.) have direct analogs in the CMS Framework
- There was a significant emphasis on fine-grained “provenance” tracking at the level of individual event products: module parameters are labeled as tracked/untracked and “tracked” parameters are stored in the data files.



Data persistence/EDM



- The CMS data persistence through 2004 had moved beyond Objectivity, but the implementation at that time (POOL with ROOT files as a backend) had many of the same problems:
 - Data for any given dispersed in many files, leading to difficulties for access and distribution (due to run-time bookkeeping layers needed)
 - Poor match with analysis requirements for Ttree-like format (to use in ROOT)
- From 2004 we reimplemented the persistence/event-data-model.
- The implementation is different (in terms of code) from BaBar, of course, but the design goals were more or less the same (next slide)



Data persistence/EDM



- Data is stored in ROOT TTree format (in principle solves the “ntuple problem”)
- References are only valid in the context of an event
- “Metadata” Trees on the side store additional information for use in standard event processing application
- The initial implementation only supported reading from one input file at a time, but we have recently added the ability to simultaneously read from two files (e.g. RAW and RECO). We do not have pure “pointer collections” as in BaBar.
- No explicit transient/persistent layer, some decoupling since genreflex/gccxml is used to provide data dictionaries (which can then be fed to CINT)



ROOT improvements



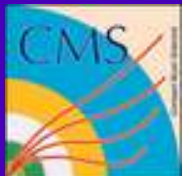
- There have been a number of improvements to ROOT. In the area of persistence, memory-use, I/O:
 - Functionality was added to do “fast-cloning” of TTree's, allowing merging (or copying from input to output) of entire trees without decompression/unpacking/packing/compression
 - TTreeCache – uses the knowledge ROOT has of the required branches and the location of the associated buffers within the file to aggregate and thus optimize the file reads themselves
 - genreflex/gccxml(/cintex) – dictionary generation
 - A variety of memory use fixes



Conditions



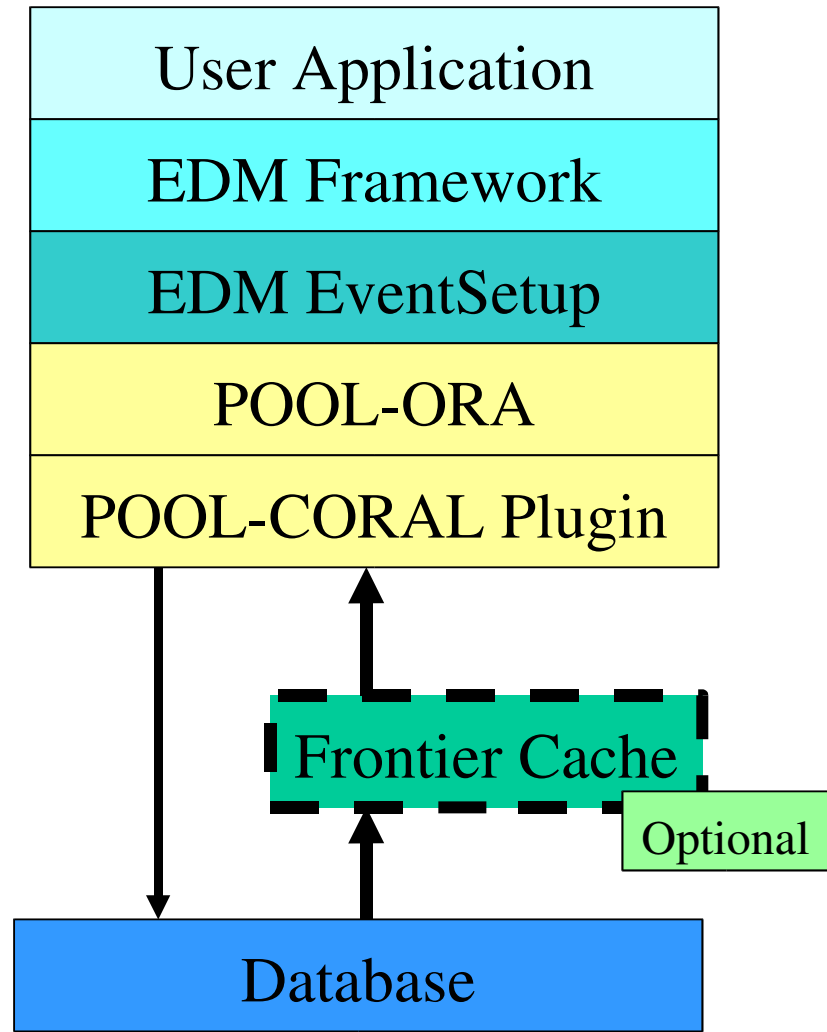
- CMS has a conditions/calibration model similar to BaBar's (with IOV's, tags, etc.)
- Conditions in a distributed environment seems to be handled in different ways for different experiments:
 - Simple flat files (Aleph, CMS - using SQLite files for testing)
 - Database
 - With remote access to central DB (early CDF Run 2)
 - BaBar Objy, but with explicit DB file copying to sites
 - With a central DB, but with caching layer at sites (late CDF, CMS)
 - Hybrid – DB with exported flat files, BaBar upgrade(?), Phenix (IIRC)

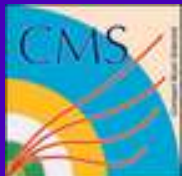


Conditions Access



- All conditions stored in Oracle
- POOL-ORA (Object Relational Access) is used to map C++ objects to Relational schema.
- A FroNTier/POOL (CORAL) plugin has been developed to enable a middle-tier proxy/caching service

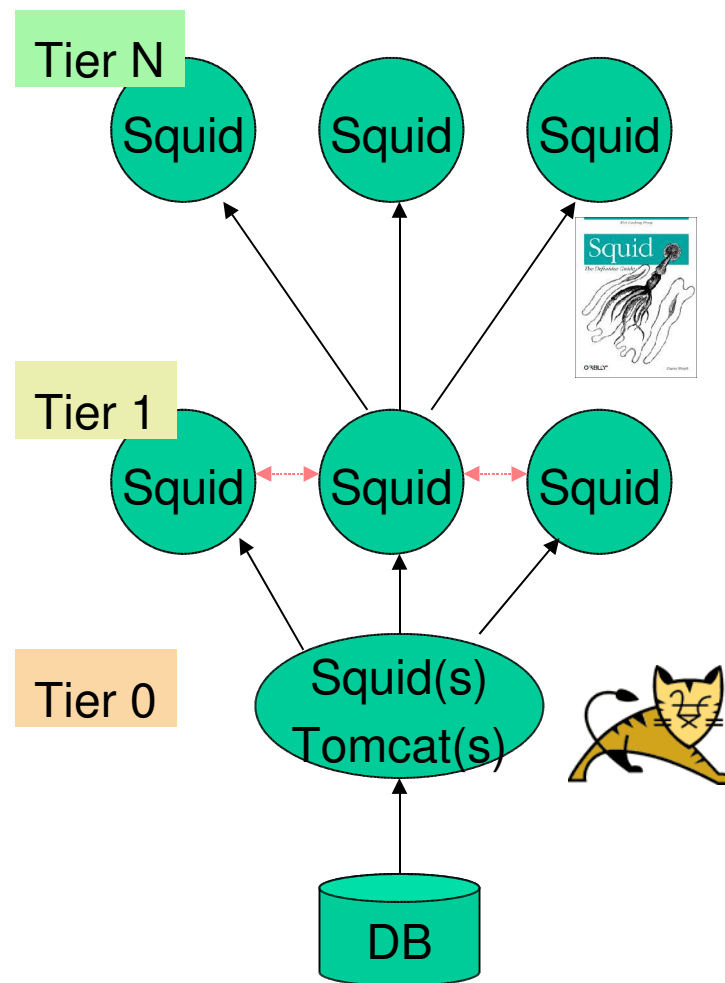




Conditions Access outside CERN



- Redundant Tomcat + Squid servers are deployed at Tier 0.
- Squids are deployed at Tier 1, and Tier N sites.
- Configuration includes:
 - Access Control List (ACL)
 - Cache management (Memory and Disk)
 - Inter Cache sharing (if desired).
- Applications needing access to conditions data are configured with a list of servers, and proxies.





Distributed Computing



- The model in CMS is slightly different than in BaBar
- BaBar:
 - SLAC – general analysis use w/logins, initially PromptReco (then Padova), skimming, etc.
 - Tier-A – skimming, analysis use w/logins
 - Tier-C – MC production, ad-hoc analysis use
- CMS:
 - Tier-0/CERN – PromptReco/calib + CAF (analysis w/logins)
 - Tier-1 – ReReco + organized skimming (no login, except FNAL)
 - Tier-2 – MC, general analysis use, no logins



Interesting topics



- From this point on I'm talking about things that CMS doesn't actually have at the moment, but which either (a) find interesting or (b) are topics we (and perhaps other LHC experiments) are pursuing



Performance tools



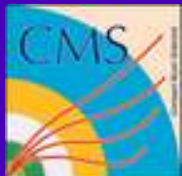
- valgrind massif – heap profiler, as of valgrind 3.3.0 this has become a useful tool: provides snapshots of heap memory in a readable format showing stack traces in a tree-like form
 - does not require code instrumentation, but slow like valgrind memcheck
- igprof – CMS in-house developed tool providing gprof-like performance numbers, even with shared libraries. It also can measure dynamic memory use (like Great Circle, say) and report it, by function, in a gprof-like format.
 - does not require code instrumentation
 - works via LD_PRELOAD, also works with BaBar code (I've tried it)



Performance tools



- [oprofile/perfmon2/perfctr](#) – modern cpu's have built-in low-level performance counters for monitoring memory cache stalls/misses, TLB misses, etc. These tools provide access to them from user space.
 - At least the latter two require kernel patches. These provide very useful information as modern superscalar cpu's, with multi-level caches, are more complex than the naive model most have in mind. The difficulty with these tools seems to be in interpreting them and translating that into code changes that improve performance. CERN has some people working on this. See this presentation for a preliminary study in CMS:
<http://indico.cern.ch/getFile.py/access?contribId=8&sessionId=0&resId=0&materialId=slides&confId=15918>



Performance counters

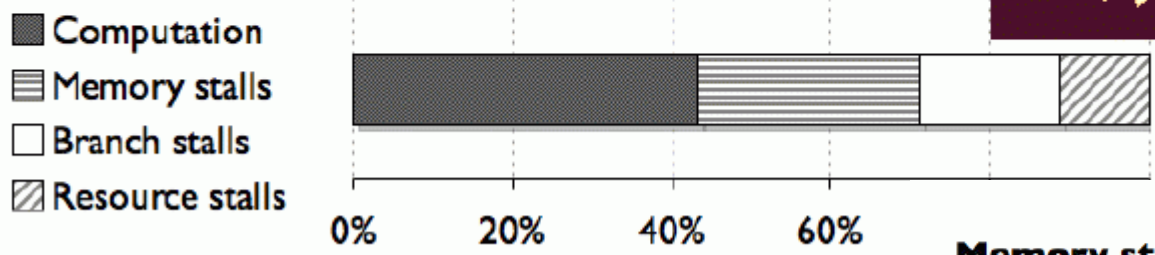
Basic characteristics – Opteron 246

42 QCD event reco job from 1.3.0 release validation

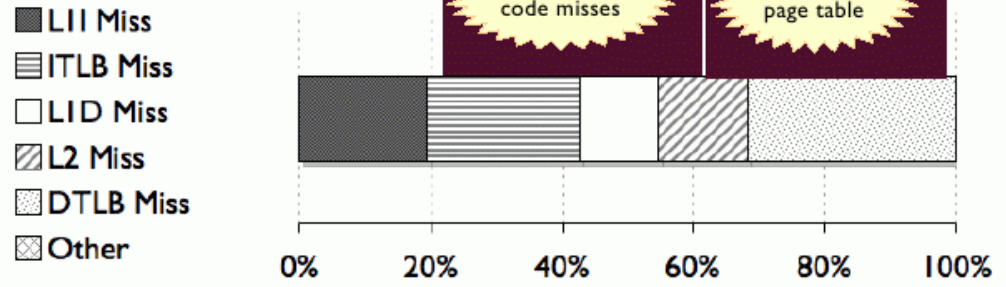
Detail remains to be improved!

From Lassi Tuura

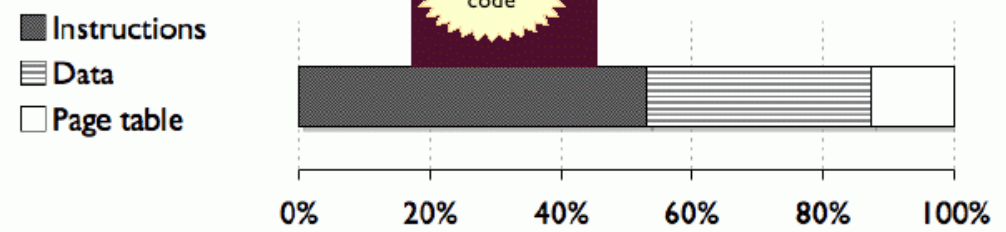
Cycle capacity use estimate (%)



Memory stall analysis



L2 cache accesses



Examples from a older CMSSW software release. (derived from oprofile, IIRC.)



Native 64-bit builds

- Clearly newer processors are 64-bit. The much touted increased address space is arguably irrelevant for HEP applications, but clearly they have other advantages in terms of performance:
 - Additional (and wider) registers, scratch registers
 - Reduced function prologue/epilogue (call overhead)
 - Instruction pointer relative addressing (e.g. reduces -fPIC cost, etc.)
- In the CMS test 64-bit builds, I've seen 20-25% performance improvements
- There is a cost: the memory footprint more or less doubles as-is
 - Some large fraction of this is due to the heavy use of shared libraries in CMS. By default the linker/loader in linux adds a 1MB page/library. This can be resolved in numerous ways, after which I estimate that the true memory footprint increase is probably around 25%.



gcc4.x/linkers



- gcc4.x of course added a number of interesting things on the performance front, e.g.:
 - They reimplemented the framework for doing optimizations so that they can be done on the intermediate Tree SSA format used by the compiler
 - There has been some amount of work on auto-vectorization of loops
 - Additional support was added for managing symbol visibility in shared libraries
- The one large change in gcc4.x was from g77 to gfortran
- Recently a new, alternate linker called gold has been added to binutils (elf-only, though) whose focus was on linking speed. The reported improvement in linking times is o(5-6).



Shared Libraries



- CMS (and all of the other LHC experiments, I think) uses shared libraries much more heavily than BaBar, both as normally linked shared library archives and as dynamically loaded plugins (for FWK modules/services, I/O plugins, etc.)
- Big advantage: development write-code/compile/link/test cycle is much faster
- But it appears we have stepped at some level into the problems which arise when the number is large (or size is small, say)
 - Performance issues (code bloat), symbol visibility issues, etc.
 - This comes from the fact that we map the physical (package) structure of the software release into a set of libraries/plugins
- We are now looking at “large” libraries and static binaries again



Multi-core processors



- In order to improve price/performance/watt the days of increasing CPU clock frequency are over, the trend now is towards multi-core cpus (4-core today, many more in the not too distant future)
- The fact that our applications are embarrassingly parallel made our initial use of multi-core cpus easy: we just treated them as separate cpus (in terms of how we configure our batch systems) and the only issue was enough real memory/core
- But if the number of cores becomes very large, this model will probably break down. Naively scaling the system memory/core and CPU memory cache effects will (probably) make this costly and poorly performant.



Multi-core processors

- For many software industry applications, the solution is simply multi-threading. Not always easy, but this is the direction things are heading.
- For HEP applications one can imagine many types of solutions:
 - Multi-threading at the FWK level on events (or “paths), but this is not easy given our SW development model (everyone must be thread-safe)
 - “Shared memory” for conditions/geometry and other similar things, this helps system memory/core, but not CPU caches issues
 - Fine-grained threading with specific CPU-costly algorithms (either custom or OpenMP-style), e.g. Geant4 is pursuing this, ROOT is pursuing various things (e.g. decompression/compression on threads). The difficulty here is balanced utilization of cores.



Conclusions



- I'm not sure I have any conclusions yet....
- In the past few years we have at some level caught CMS up with some of the things that BaBar/CDF/CLEO-III/c had learned and made some steps forward in some areas
- Still lots of interesting things to do, however
- And everyone is waiting for LHC to start to see which new and exciting problems arise....