9.6 Data Analysis and Programming Environment: Distilling Information

René Brun

Data storage formats and analysis tools

Data from bubble chambers and optical spark chambers were recorded on photographic film. Points on selected tracks were measured on the film and stored on magnetic tape as "raw data" in relatively simple formats specific to each experiment. Raw data from the early experiments using digitized spark chambers and simple calorimeters were also stored in experiment-specific formats. Then the reconstruction program was processing these events and writing a "Data Summary Tape" (DST) with compact data types for the reconstructed tracks (direction, momentum) or calorimeter information. The next step was data analysis: The task was to assign or calculate particle masses and other quantities of interest, store them and display selected information in histograms. During the 1960s, the operation was optimized for the bubble chamber experiments by producing a suite of programs THRESH, GRIND, and SLICE, implementing the three stages of geometry, kinematics and assignment of mass to each particle: they could be used by all experiments with little customizing [45]. During the early 1980s, it was realized that this process could be simplified by storing the quantities of interest in "n-tuples" (a simple row of n variables put into tables). The Physics Analysis Workstation PAW [46] system developed in 1983 was based on row-wise n-tuples. A user could interactively process the n-tuples data with simple queries. Histograms were automatically viewed in many possible graphics formats on the user workstation. The system became rapidly very popular for data analysis and n-tuples grew larger. PAW n-tuples became the main analysis format for most experiments at CERN and other labs between 1988 and 1998.

In 1995, the ROOT project was started based on the same philosophy as PAW, but implemented in C++ and able to support more complex data types [47]. By 2000, ROOT had a built-in system capable of accepting most C++ constructs found in the LHC experiments. ROOT had been selected in 1998 for the Tevatron experiments at Fermilab, followed in 1999 by the RHIC experiments at BNL. All LHC experiments, as well as most HEP or related laboratories in the world adopted the ROOT system. ROOT was progressively interfaced with many other tools (graphics, user interfaces, statistics, mathematical libraries, etc.), becoming de-facto the new CERN Program Library. About 20,000 people download the ROOT binary or source files every month and about 9,000 people are registered to the ROOT Forum. Besides being used by more and more people in the scientific domain, ROOT is also used in the car and oil industry, and in finance.

Programming Languages

For about four decades, FORTRAN was the main programming language. Simple to learn and well adapted to small or medium size single process and sequential applications, it was the language of CERNLIB, the CERN Program Library [48], built from contributions of users developing simple algorithms, linear algebra or general utilities with up to a few hundred lines of code. CERNLIB was very popular in the scientific community: during the 1970s, it included task-oriented packages with a few thousand lines of code, e.g. GD3 for basic graphics, HBOOK [49] for histogramming, MINUIT [50] for fitting and minimization. At the same time data structure management systems were developed. These contained both numerical information and logical information about the object they describe, and included HYDRA [45] for bubble chamber experiments and ZBOOK [51] for electronic experiments. The ZEBRA system (ZBOOK+HYDRA) appeared in 1982, enabling the creation and management of a complex network of dynamic data structures, and saving them to files in machine independent format.

With the advent of FORTRAN90, it would have been possible to upgrade ZEBRA, taking advantage of the new features (similar to C structures), but this did not happen: FORTRAN was less and less competitive with the new software programming systems that provided user interfaces, graphics, network communications and interfaces with the operating system and hardware devices. With the increasing number of personal workstations and network connections, more and more applications required calls to the operating system, file and network system. These interfaces were typically implemented in the C language. For example, in the PAW system a substantial fraction of the user interface and graphics was in C.

The years 1990–1994 witnessed many discussions about the future language for HEP. By the end of 1994, it was agreed that the programming language for HEP was going to be C++. Two Research and Development projects were launched: RD44 for the development of Geant4 and RD45 for the evaluation of Object-Oriented data bases. Starting in 1995 the ROOT system also opted for C++.

Programming Environment

The size of programs has grown with time at a tremendous rate. In the early 1970s, large programs were around 10,000 lines of code. CERNLIB was some 30,000 lines of FORTRAN and assembler in 1970. The HBOOK or HYDRA packages contained around 15,000 lines in 1975. CERNLIB reached a maximum of about 150,000 lines in 1985, including the major packages. ZEBRA stabilized with about 30,000 lines in 1985. Geant3 started with ~10,000 lines in 1981 and reached ~200,000 lines in 1993; PAW reached a maximum in 1994 of ~250,000 lines. During the 1970s, an experiment-specific code (essentially the reconstruction

program) was less than 10,000 lines, reaching about 100,000 lines at the start of LEP in 1989. Today, Geant4 contains ~1.5 million lines, ROOT ~3 million lines, with specific software on top of these packages weighing in at ~5 million lines per experiment — not counting the analysis software by thousands of physicists. The memory used by these programs was around a few tens of kilobytes in the 1970s. The full simulation of the OPAL detector planned for LEP in 1981, including electromagnetic and hadronic showers, could run on central computers having only 1 MB of memory. From LEP to LHC experiments have grown in size and complexity, and the signal to background ratio has become much more demanding: today one sees simulation systems requiring 3000 times more memory! (Fig. 9.8)

Managing the source code of HEP software has always been a concern, even when systems had only a few thousand lines of code, because the software development has always been a joint effort in experiments. The PATCHY system [52] was developed at the end of the 1960s for bubble chamber software. A PAM file (PATCHY Master file) was common and read-only for all developers. Each developer prepared his/her own file with changes or additions to the master file. A PAM file was updated periodically by user consensus. PATCHY remained the preferred solution in the 1980s until an interactive version of PATCHY, called CMZ, appeared in 1989, making it easier to develop software on distributed systems by using workstation with nice text editors. PATCHY/CMZ were replaced

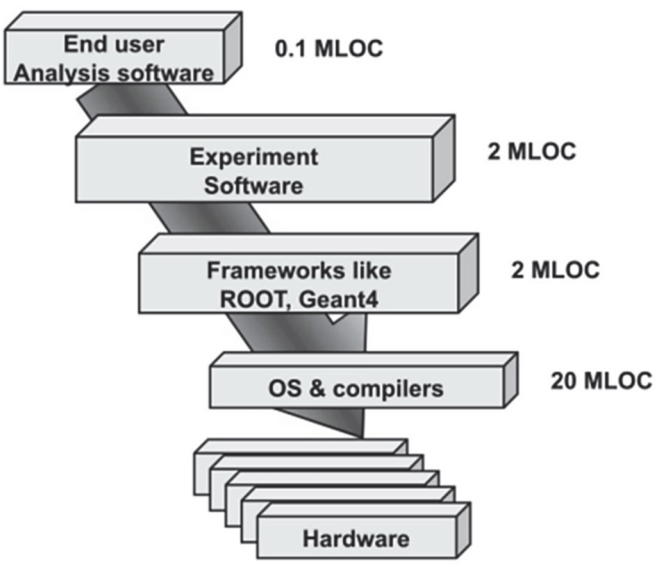

Fig. 9.8. The Software hierarchy (MLOC ↔ Million Lines Of Code).

by the open source CVS (Concurrent Versioning System) appearing at the end of the 1990s. With CVS, it became possible to edit/manage a central code repository shared by hundred/thousand developers, showing differences between successive versions and making easier code releases. Around 2005 an evolution of CVS, called SVN (Sub Version System) was adopted for all developments in LHC computing (and everywhere else). The GIT system replaced SVN around 2012, offering improved capability to manage successive versions of code developed and used by thousands of people. This trend will likely continue, following the continuous requirements to manage many millions of lines of code.

Error reporting systems have also evolved in the wake of the growth of the software size. From pure oral reports in coffee meetings or paper mail, huge progress followed the installation of the local area CERNET, followed by BITNET in the early 1980s, which brought worldwide e-mail exchange. The first web-based systems appeared around 1995 allowing the visualization of the source code and its documentation. The first common error reporting and managing system SAVANNAH appeared in 2000, followed by the current JIRA system in 2012. Several projects also host a discussion forum where users can report questions and problems, so that more experienced users can help (e.g. the ROOT forum).

Outlook

Computing at CERN has followed and continues to follow the general trends in the field. The computing languages C++, Python, Java are now the main languages in science, industry, communication and information. This approach facilitates communication among these various fields, but is also a important asset for the careers of young scientists who seek employment outside High Energy Physics (HEP).

General tools like ROOT will continue to evolve following the general trends not specific to HEP like user interfaces, graphics and networking. A considerable effort is devoted to integrate machine learning libraries and techniques to facilitate the ever more complex data analysis phases. The Open Data philosophy facilitating the use of experiments data for teaching will also require simple and common user interfaces, professional documentation and data descriptions.

The detector simulation with the Geant family will have to evolve to cope with the high luminosity machines and increasing energies. Factors of 10 to 100 in processing speed will have to be gained, taking advantage of parallel architectures and by integrating fast simulation algorithms into the full and therefore necessarily slow simulation system. Machine learning techniques will enter this field to optimize automatically the simulation and analysis processes.