Concluding remarks: Emerging topics

D. Perret-Gallix[a*]

[a]LAPP - 9 Chemin de Bellevue - BP 110 F-74941 Annecy-le-Vieux CEDEX - FRANCE

In summing up this workshop, we would like to open a broad discussion on additional emerging topics that may contribute to shape the future of physics research computing activities. To initiate this global discussion let me address in this short contribution some of these issues: distributed public computing, social or collaborative software, web computing, high precision numerical computation, common development platforms and languages issues. We welcome contributions to this discussion on the ACAT Twiki web site.

1. Introduction

Vivid presentations were given, new collaborations started, a rich harvesting of high quality contributions presenting a snapshot of today computing in physics have produced valuable proceedings.

An eclectic selection of exciting plenary talks from historical perspectives of physics in Berlin to today DESY research, from frontier development on quantum computers to AI advances on symbolic manipulation language and neural nets was presented. For the first time in this workshop series, high precision numerical computation issues motivated by demanding physics analysis at the new colliders were addressed.

But, at the end of this meeting, we are also left with vague but lingering feelings that we may have missed or overlooked important emerging developments; that we may have neglected to bring to the fore controversial issues, that new needs and concerns were not properly addressed. This research field is becoming so complex that no one can grasp it in all its dimensions. A collaborative effort is needed to forecast and assist the development of this fast changing field and to prepare the next workshop programme. You are all invited to share your vision and expertise on the ACAT web site[2].

Let me start here this exercise by addressing essentially 4 topics: distributed public computing; social, collaborative software and web computing; languages issues and high precision numerical calculus.

2. Distributed computing: the performance race

Modern physics research relies nowadays on three components, vertices of a virtuous triangle: experiments, theory and computing. In the long history of research, computing is the "new kid on the block". There are no experiment design, no data taking, no data analysis, no data interpretation or no theoretical prediction without computing. Computing has become ubiquitous in physics research.

But computing has also to deliver or it would be seen as an impediment to the progress of science. It has to fulfill the increasing science requirements that, in term of performance, have become gigantic, much larger than what "Moore's law" allows. The only alternative is to parallelize or distribute the computing load over a large number of processing units.

Massive parallel systems, computer clusters, data/compute GRID architectures or dual/quad/multi-core processors have attracted most of the large investments both in human resources and financial support. If many aspects of the GRID projects were properly covered in this workshop, little was said about another approach to distributed computing: **"Public or Internet Distributed Computing" (PDC)**.

*Denis.Perret-Gallix@in2p3.fr

[2]acpp.in2p3.fr/cgi-bin/twiki/bin/view/ACAT/WebHome

Several scientific projects based on the **BOINC** (Berkeley Open Infrastructure for Network Computing) package released in 2003 are now running routinely on hundred thousands of personal PCs.

The BOINC paradigm is to make use of the CPU cycle, memory and disk space donated by public users or companies. During system idle time, a client code selected by each individual's own scientific interest can be set to run. The computing power hence provided would be limited, but when hundred thousands or millions of Pcs are ganged together, the resulting power is equivalent to several No.1 Top 500 mainframes.

The quality factor of the PDC is not as high as what one can expect from a dedicated mainframe. Due to computer failures, malicious interferences or slow networks, some computing redundancy must be built-in, resulting in a reduction of the pure expected power. Nevertheless the potential is huge and growing as shown by the **SETI@Home** pioneering project which in 2004 was reporting using 1 million user computers for a sustained processing power of 70 Tflops. At this time there was roughly 300 million hosts connected to the Internet. This small fraction (0.3%) can be certainly increased by giving the project a wider publicity and more incentives to the users. In the longer term, the number of computers in the world is expected to rise up to 1 billion by 2015 so that scavenging Pflops is not out of range.

In return each user receives credits proportional to its PC computing time and a "hall of fame" ranking is shown on the web. But a very important motivation both for physicists and for the users is the invaluable outreach that comes with it. Through the screen-saver displaying the calculation status and web pages presenting a lively description of the science research being pursued, the user gets the feeling of becoming part of this great scientific endeavor.

Since the introduction of the open source BOINC packages, this approach has been followed by climate prediction, protein structure, gravitational wave physics researches. CERN has based on it its long term simulation of LHC beams. But many other applications can be developed in HEP that would benefit both the research needs and the public information.

3. Social, collaborative and web computing: better communication

After pointing out the necessity to increase strongly the computing power open to scientific research, let me address the next most important issue, namely: more and better collaborations. As said previously, our field progresses thanks to the expertise of computer scientists, physicists, theorists and mathematicians. But each group has different motivations, different thinking patterns, uses different concepts expressed in different languages. Moreover our community is large and spread out over distant places.

Therefore surveying, discussing and, possibly, developing tools to bridge these breaches has been a permanent concern in this workshop series and should remain high on the agenda. These questions have triggered both in the academic and commercial worlds the development of tools which include email, chat, net-conference, VoIP, groupware or blogs.

Some of these technologies rely on the development of additional middleware layers between the browser and the server like **AJAX** (Asynchronous JavaScript and XML) to speed up web applications. More and more "office applications" used to run on isolated computers can be ported to the web to create a new collaborative environment.

Simultaneously, a new revolutionary use of the web is taking place based on the **Wiki** concept. Even if the main ideas have matured for more than a decade, the Wiki technology is changing the way we use the web. It gives the possibility to everyone, would it be physicist, theorist or mathematician, to modify or add new text or file to any Wiki web page using simply a web browser.

This may not appear, at first sight, as a major innovation, but it brings solutions to many every day problems. A documentation or information web page can be updated or created by the very author of the news without relying on a webmaster. This is a road to an always up-to-date information. Everyone can talk to everyone without going through mailing list discussions. Scien-

tific papers can be written quasi-simultaneously over the Internet without the cumbersome cycle of comments and corrections through email. A common global knowledge database is being progressively built.

The success of **Wikipedia** is here to show that this model works and provides a new communication technology that can be used both at the global level and within a limited collaboration. The selection of the most appropriate Wiki implementations, the development of specific tools like latex-ps-pdf interfaces or the implementation of collaborative code development are important issues.

Collaborative developments need also better code and project documentations. Automatic documentation systems digging comments and information from the code files themselves are a great improvement toward an on-line instantaneous project documentation. **Javadoc**, **Doxygen** or **Root THtml class** are good examples of such packages.

Collaborative projects are more easily managed if a common development platform is used by all contributors. Among them, the **Eclipse** integrated development environment, based on Java, runs on most OS and computer systems. Many computer languages, modeling and documentation tools have been implemented at various level of interoperability. Initiated by several major companies, it is now made available to the open source community. Comparisons with other potential packages like **Xcode**, **Kdevelop** or **NetBeans** as well as possible HEP specific plugins developments should be addressed.

Finally, software engineering has been, so to say, revisited and a "new" concept called by the fancy name of **"extreme programming"** or XP has made its way to the community. XP is "a deliberate and disciplined approach to software development". This approach precisely insists on: setting up short links between programmers and users, making short development cycles but with many iterations, promoting strong involvement of end-users and pair programming (one workstation for two persons). Is this model appropriate to physics research activities and to the new societal environment? In any case this is pushing to the extreme the obsessing need to develop and support collaborative work, trademark of this workshop series since its origin.

4. Language issues

A major issue related to collaborative developments is the selection of the programming language(s) supporting common projects. In the workshop series, language issues have received a lot of attention. Progressively this field of research has drifted from FORTRAN only computing to object-oriented programming (OOP). C++ and, for some applications, Java have become the workhorses of the new physics research computing activity. However, some of the physicists used to contribute strongly to simulation, data analysis or even on-line programming have been lost in this process as the learning curve is much longer than for FORTRAN. This is a typical example of the societal changes mentioned in the previous section.

The computing technology is becoming so heavy that, like mechanical construction already two decades ago, it is now becoming a "chasse gardée", a property, sometimes unwantedly, of computing experts. This may prove to be inefficient in the long term. Any physicist (accelerator expert, experimentalist, theorist) should be able to implement his/her own ideas or conceptual work into a practical code to be used by the rest of the community. I do not believe that outsourcing code development will benefit our community in the end.

In the meantime FORTRAN has evolved from FORTRAN 77, to 90, 95 and 2003. This last standardization has been designed to produce an advanced OO language still retaining its numerical computation performance and capability (quadruple precision for some compiler), highly parallelizable code and optimized numerical libraries. In addition, it has a quite efficient C/C++ interface.

Without embarking on a "religious war", one should re-examine the language issue in the light of its social acceptance aspect, probably in the framework of a mixed language environment. Also related to this discussion are the benefits

and trade-offs of compiled and interpreted languages. Interpreted languages, being "interactive", are easier to learn and to use than compiled languages, but lack execution efficiency and rapidity. The more technical issue is what are the benefits to use interpreters built on genuine compiled languages like **Cint/Root** or **Ch** for C++ instead of compiled version of interpreter-born languages like **Python** or **Ruby**.

5. The 5^{th} dimension of computing power

The performance of a computing hardware is usually described by a) the CPU/bus clock frequency, b) the size of the instructions dictating both maximum memory size and the number of parallel execution units, c) the size of the memory and caches d) the speed of the interconnection with other CPU's for parallel system. This is essentially a 4-D optimization problem. However, there is a 5^{th} dimension, somewhat overlooked: the floating-point unit (FPU) bit size.

Available both on 32 and 64 bits CPUs, the double precision (64/80 bits) FPU provides an accuracy good enough for most applications, but **quadruple precision** capabilities are becoming more and more important for many research domains, not only in high energy physics but also for non-linear process simulation, number theory and also for commercial applications like finite element modeling CAD, 3-D real-time graphic, statistics or security cryptography.

High precision computation not only gives more precise results but, when large number cancellations or rounding errors play an important role, leads more surely to the correct results. In addition, high precision improves the convergence of some iterative algorithms (like linear equation solving systems) reducing the number of necessary iterations. This is virtually equivalent to an overall computational speed increase.

Unfortunately the gain is compensated by the additional computing time imposed by the software emulation of higher precision computations. Conventional libraries implementing quadruple precision floating-point in C++/F90 show performance drop by a factor 10-20 compared to double precision for basic operations on PC. The Linpack benchmark jumps by a factor 36 when converted to quadruple precision.

Providing hardware 128 bits FPU precision would result, then, in a substantial computing time reduction and therefore in a better FLOPS figure. In these cases, numerical accuracy is really the 5^{th} dimension of computer system performance.

This is a very complex issue that has triggered the setting up of a pluri-disciplinary working group to propose the best software/hardware balance for achieving high precision numerical computation (quadruple/octuple). The conception and proto-design of a dedicated co-processor under the proposed acronym "**HAPPY**" for "High Arithmetic Precision Processor Yoke" is part of this study.

Many numerical calculations would get a dramatic improvement and the co-processor would become an important asset in the Petaflops race.

6. Conclusion

This contribution which has largely benefited from discussions with several colleagues during and after the ACAT05 workshop is an invitation to a more thorough and meaningful survey that only a collaborative effort can deliver. We expect anyone interested on this forecast to contribute to the forum open on the ACAT TWiki page. Not only your contribution will be central to the organization of the next workshop in 2007 in Amsterdam, but will in many respects contribute to the successful completion or design of the upcoming physics projects.