# MASSIVE TRACKING ON HETEROGENEOUS PLATFORMS

E. McIntosh, F. Schmidt, CERN, 1211 Geneva 23, Switzerland
Florent de Dinechin LIP, ENS, 46 allée d'Italie, 69364 Lyon cedex 07, France

*Abstract*

The LHC@home project uses public resource computing to simulate circulating protons in the future Large Hadron Collider (LHC). As the motion of the simulated particles may become chaotic, checking the integrity of the computation distributed over a heterogeneous network requires totally identical floating–point behaviour, regardless of the type of computers used. This paper reviews the problems encountered and the practical solutions.

## INTRODUCTION

At CERN, the structures and priorities of the installed computing facilities have always been dominated by the needs of the experiments. This is still true today with the development of the LCG, the Worldwide LHC Computing Grid. In 1992, however, the PaRC cluster of IBM workstations was established to meet the needs of the engineering community, and then in 1997 when the LHC Machine Advisory Committee recommended a significant increase in tracking studies, an informal Numerical Accelerator Project (NAP), was established to provide dedicated computing capacity. The first installation was a ten processor Digital TurboLaser with 800 CUs, [1] later augmented by ten Digital workstations to provide more than double the capacity. More recently, in line with industry trends and CERN Information Technology policy, this was all replaced by sixty–four 2.4GHz dual processor PCs, providing over 50,000 CUs, and operated on a fair share basis as part of the central Linux batch service.

A typical LHC Dynamic Aperture (DA) study can analyse 5 angles in phase space, 60 random representations of the magnetic errors called seeds, and a certain number of phase space amplitude ranges; each case tracking 30 particle pairs for $10^5$ turns. Such a study requires a few thousand independent jobs of approximately two hours CPU each, and can thus normally be completed in a matter of days.

Half of the NAP capacity was then allocated to Beam Collimation studies but at the same time there was a desire to perform a tune scan with beam to beam interactions for one million turns. This would require some six million CPU hours and was clearly not feasible on the existing cluster [1].

[1] A CU, CERN Unit, is defined as the single CPU power of an IBM 370/168 or Digital VAX 8600 computer on a set of four physics benchmark programs.

## THE CPSS PROJECT

The CERN Physics ScreenSaver system (CPSS) [2] was established to use the unused CPU cycles of the some 5,000 WINDOWS desktop PCs at CERN, a CERN investment of several million dollars. Assuming a very conservative 50% idle time and 50% powered on, this distributed computing power could provide an order of magnitude increase in the available CPU capacity. The SixTrack program [3] has modest input/output requirements of less than 1MB/6MB (50KB/2MB when compressed) and has a working set/memory requirement of 32/65MB and was an almost ideal application for distribution over a network; it is portable and part of the SPECFP2000 benchmark suite [4].

A.Wagner of CERN provided a WEB application containing the job repository, a lightweight screensaver for WINDOWS and a set of PERL routines callable from LINUX, for task group creation, task submission, result retrieval and status enquiry (see Figure 1). A SixTrack Checkpoint/Restart facility was introduced to allow frequent instantaneous screensaver interruptions without compromising the efficiency of long–term runs. First tests were carried out using a Compaq Fortran compiler, the only Fortran compiler available on WINDOWS at CERN. After resolving a couple of minor issues by dummying out the CERN Library graphics package and removing the Line-Feeds in the WINDOWS text files, the first major obstacle was found to be the processing of NaNs and Infs, which was up to 1000 times slower than normal. As testing and branching are relatively expensive operations compared to arithmetic, the application checked only once per turn for lost particles and the code was modified to check at every element of a turn. Several hundred CERN users volunteered their desktops and a first 1500 task study was carried out successfully, with the results for the average DA being within 3 parts in a thousand of the Linux batch results. Investigation of the differences found that about 1000 of 60 million input parameters were being converted with a one unit in the last place (ULP) difference between WINDOWS XP and WINDOWS 2000. Purchase, installation and usage of the Lahey–Fujistu Fortran 95 lf95 compiler as used on Linux removed this problem. Further testing then seemed to show identical results from Linux and WINDOWS. This should perhaps not be surprising since compilers could use the same code generation on either system, apart from the system interface calls. The manufacturer "strives for this compatibility" but it is not guaranteed. This was a major step forward; up to this point a study was always carried out on completely compatible processors and systems [5].

Suitably encouraged, a 300,000 task study of 1000 an-

Taskgroup List - Overview

| Id | Name | P-ID | Status | Priority | Progress | Created | Completed | # Jobs | | | | | Progress [%] | CPU total [s] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Tot (Del) | Q | A | RR(S) | Done | | |
| 62 | SixPro300 | 163 | N/A | 3 | Done | 7/17/2003 4:11:24 PM | | 262 () | 0 | 0 | 0 (0) | 262 | 100 | 1760172 |
| 63 | SixPro150 | 163 | N/A | 3 | Done | 7/23/2003 4:54:06 PM | | 414 () | 0 | 0 | 0 (0) | 414 | 100 | 8237650 |
| 64 | Sixtrack Samples I | 158 | Hold | 3 | New | 8/5/2003 6:57:49 PM | | 5012 (0) | 2234 | 0 | 2761 (17) | 0 | 0 | 447840 |
| 113 | w1_lhc_b6-m4 | 235 | Hold | 3 | Started | 10/14/2003 1:34:38 PM | | 1817 (0) | 0 | 0 | 0 (2) | 1815 | 99.89 | 9039574 |
| 114 | Sixtrack NC II | 160 | Hold | 3 | New | 11/6/2003 12:40:26 PM | | 3000 (0) | 1 | 0 | 2208 (791) | 0 | 0 | 1460494 |
| 115 | wdesk_lhcdesk | 236 | Hold | 3 | Started | 12/7/2003 1:12:05 PM | | 55 (1) | 0 | 0 | 11 (0) | 44 | 80 | 293108 |
| 116 | wdesk_lhcdesk60 | 237 | Hold | 3 | Started | 12/8/2003 4:32:48 PM | | 6047 (8) | 0 | 0 | 0 (5) | 6042 | 99.92 | 14489977 |
| 117 | wdesk_lhcdesk9 | 238 | Hold | 3 | New | 1/14/2004 4:21:22 PM | | 0 (0) | 0 | 0 | 0 (0) | 0 | 0 | 0 |
| 118 | wdesk_lhcdesk20 | 239 | Hold | 3 | Started | 2/20/2004 4:30:06 PM | | 3040 (0) | 1 | 0 | 0 (25) | 3014 | 99.14 | 7123689 |
| 119 | wdesk_lhcdesk10 | 241 | Hold | 3 | Started | 3/8/2004 10:33:20 AM | | 304 (0) | 0 | 0 | 0 (1) | 303 | 99.67 | 449144 |
| 120 | massimo_lhc.D1-D2-MQonly-inj-no-skew | 242 | N/A | 3 | Done | 3/12/2004 3:41:45 PM | | 1651 (8) | 0 | 0 | 0 (0) | 1651 | 100 | 5557450 |
| 121 | massimo600_lhc.D1-D2-MQonly-inj-no-skew | 243 | Active | 3 | Started | 3/13/2004 3:54:40 PM | | 18535 (319) | 0 | 2 | 6 (56) | 18471 | 99.65 | 51482172 |
| 125 | eric_eric20 | 246 | N/A | 4 | Done | 4/28/2004 3:27:07 PM | | 600 (0) | 0 | 0 | 0 (0) | 600 | 100 | 1245 |
| 126 | eric2000_eric2000 | 247 | N/A | 4 | Done | 4/28/2004 4:59:11 PM | | 601 (0) | 0 | 0 | 0 (0) | 601 | 100 | 1237 |
| 130 | massxp_massxp | 254 | Active | 3 | Started | 5/10/2004 4:29:57 PM | | 1875 (1852) | 0 | 0 | 9 (14) | 0 | 0 | 5575727 |
| 131 | mass20_mass20 | 255 | Hold | 3 | Started | 5/10/2004 4:53:14 PM | | 1601 (1583) | 1 | 0 | 17 (0) | 0 | 0 | 5061200 |
| 132 | masslahey_masslahey | 256 | Active | 3 | Started | 5/13/2004 7:54:33 PM | | 7433 (0) | 0 | 3 | 0 (2) | 7428 | 99.93 | 27978702 |
| 134 | massxp_massany | 259 | Active | 5 | Started | 5/18/2004 6:37:42 PM | | 1838 (0) | 0 | 0 | 7 (25) | 1806 | 98.26 | 5554755 |
| 135 | w1_lhccpss | 260 | Active | 4 | Done | 5/28/2004 6:00:45 PM | | 20 (0) | 0 | 0 | 0 (0) | 20 | 100 | 606 |
| 136 | w1_lhc1000 | 261 | Active | 3 | Started | 6/2/2004 2:03:10 PM | | 6274 (0) | 2142 | 75 | 1 (6) | 4050 | 64.55 | 17450578 |

Total CPU Usage: 201346226 (2330 d 09 h 30 m 26 s )

Figure 1: An overview of the CPSS TaskGroups during the early development

gles for $10^5$ turns was initiated in order to prove the long–term efficiency and reliability of CPSS while generating some useful results with respect to the resulting DA and initial angle [6]. In parallel, testing of the more time–consuming and floating–point intensive beam–beam cases commenced. Once again a few tasks per thousand gave different results after a million turns. This was quickly discovered to be a difference between 32–bit and 64–bit systems (Intel IA–32 and Intel IA–64/AMD Athlon 64). Having verified identical initial conditions, ensuring that the same accelerator was being simulated, exhaustive analysis traced back to the turn where the difference first arose, and then to the relevant step within the turn, to the relevant statement in the step, and finally to the actual floating–point operation itself. The analysis required comparison of the actual binary representations of the floating–point numbers in order to find the first occurrence of a difference as small as 1 ULP. Computing a single particle trajectory requires from ten to a few hundred floating–point operations for each of ten thousand steps in a single turn, and even if the smallest difference is quickly magnified the difficulty is in finding the initial difference. In this case the trigger was the evaluation of an exponential function, as was quickly proven with a test program of a few lines using the exact argument in question. A similar difference was traced to the natural logarithm function. Indeed, in most systems these functions use x86 machine instructions such as `fyl2xp1` and `fyl2x`, whose microcode implementation give different results on different processors, for reasons exposed below.

At this point we considered abandoning the goal of exact results reproducibility and restricting a specific study to either only 32–bit, or only 64–bit, systems. Given the gradual introduction of the new extremely powerful 64–bit systems, but also the huge number of existing 32–bit systems, it seemed that they would co–exist for quite some years to come. It still seemed very desirable to be able to use either type indifferently.

## FLOATING–POINT ARITHMETIC

Floating–point portability, or rather the lack of it, has been well studied and debated for many years; see, for example, Goldberg [7], with the Priest supplement [8], or Belding [9].

For our application we limit the definition of heterogeneous to any Intel Pentium or compatible PC, at least for the time being. The IEEE 754–1985 standard, to which almost all modern processors conform, “specifies basic and extended floating–point number formats; add, subtract, multiply, divide, square root, remainder, and compare operations; conversions between integer and floating–point formats; conversions between different floating–point formats; conversions between basic–format floating–point numbers and decimal strings; and floating–point exceptions and their handling, including nonnumbers”. Of the options covered by the standard we consider only double precision arithmetic, with rounding to the nearest representable number (nearest even, if a tie).

However this standard is incomplete and open to interpretation. Strict compliance tends to conflict with performance and therefore tends to lose out in a competitive market. It also needs to be considered in the context of the relevant programming language standard. For instance the Fortran standard defines a unique order of evaluation only for a fully parenthesized expression, so portability may require some code rewriting. It will also require some explicit control of the compiler. In our case, we use the lf95 compiler which always disables *extended precision*, and in addition we specify the compiler option – –tp “generate Pentium code” so that architecture-specific features such as SSE2 or 3DNow will not be used. Finally, the current standard does not cover elementary functions such as exp, log, and the trigonometric functions. In the following the exp function will be used as an example.

## THE CRLIBM SOLUTION

The problem of providing the correctly rounded function results is known as the "Table Maker's Dilemma". If we consider the case of rounding to nearest, the problem arises when the function result lies very close to the mid–point of two adjacent floating–point numbers, and rounding an approximation to f(x) may not give the same result as rounding f(x) itself [10]. As a pathological case for double precision exp with a 53 bit mantissa, we use the example of [11] where in binary notation

$x = 1.[52]1 \times 2^{-53}$

where the number in square brackets denotes the number of consecutive occurrences of the following digit. In this notation

$\exp(x) = 1.[51]001[104]1010101...$

and the correctly rounded to nearest double precision result is

$\exp(x) = 1.[51]01$

In this particular example even a quadruple precision approximation, correct to within 1 quad ULP (113 bit mantissa) may deliver one of three results:

$1.[51]010[58]00$
$1.[51]001[58]11$
$1.[51]001[58]10$

but rounding the last result will give an incorrect answer.

Software libraries exist that increase the working precision until correct rounding can be decided. A Web search discovered several such libraries for double precision results. Given that our aim was reproducibility of results, rather than precision per se, this may seem a very extreme solution. Its advantage is that all these libraries will always agree to the last bit, so choosing one of these also ensures that any of the others would do instead.

The first of these libraries was IBM libultim [13], which is no longer supported. MPFR [12] is an arbitrary precision library, and is therefore extremely slow compared to the default libm of Linux (see Table 1). The crlibm[14] library from ENS-Lyon is well-supported, is proven theoretically to deliver the correctly rounded result, and does so with a modest decrease in performance with respect to a standard libm[14]. Lately, SUN has also been developing a correctly rounded libm called libmcr.

| | Average | Maximum |
|---|---|---|
| libm | 365 | 5528 |
| crlibm | 432 | 41484 |
| mpfr | 23299 | 204736 |

Table 1: Relative timings for the exp function on a Pentium 4 Xeon, with gcc 3.3

To illustrate the scope and magnitude of the portability issue, we use one of our test programs. A simple test of the exp function, using g77 and libm, with one million arguments between -0.5 and 0.5, found 5 differences IA–32 to IA–64, 7 differences IA–32 to AMD64, and 2 differences between the IA–64 and the AMD64. All differences are of 1 ULP in these and following results. Comparing libm and crlibm we find that libm delivers the correctly rounded result in all but 304 cases provided Extended Precision is enabled; if not, there are 134,623 cases with incorrect rounding. Similarly, but using the lf95 compiler and library we find 7 differences IA–32 to IA–64, 7 differences IA–32 to AMD64, and 4 differences between the IA–64 and the AMD64. The same test program using crlibm, and compiled with five different compilers, gave *identical results* on the three architectures.

The crlibm library in 2004 provided exp, log, log10, sin, cos, tan, atan, sinh, and cosh functions [2] in the four standard rounding modes. A few simple editor scripts sufficed to change all SixTrack function calls to their crlibm equivalent for round to nearest, exp to exp_rn, sin to sin_rn, etc.

With the modified SixTrack, all numerical differences in our tests now disappeared even in the most demanding beam–beam case, which make multiple calls to exp and log in each beam–beam interaction. The 1000 angle study was restarted with the new version and all tasks were run at least twice. The only numerical differences found were due to failing processors; two desktop machines and one batch processor in the Computing Center.

In summary, in order to generate two executables giving identical results for Linux and for Windows, the task list is as follows:

- Change all elementary function calls to the crlibm equivalent
- Download and compile crlibm with portable options [3]
- Use the Lahey–Fujitsu Fortran 95 compiler with Extended Precision disabled by default.
- Generate code for any Pentium with – –tp.
- Make a statically linked executable.

After running over one million $10^5$ and $10^6$ turn jobs, most of them at least two or three times, we are confident (although we cannot prove it) that any numerical differences are due to hardware failures or overclocked processors.

## LHC@HOME

During this period of development, CERN colleagues set up a single Berkeley Open Infrastructure for Network Computing (BOINC) server [15], a successor to SETI@home, and suggested using SixTrack as a pilot application. SixTrack was modified to call the BOINC interface routines, and to return only the results summary file

[2] The functions asin, acos, and atan2 were implemented in terms of atan, which provided portable, but not necessarily correctly rounded, results.

[3] On Windows it is necessary to use either Fujitsu C included with the lf95 compiler or gcc and CYGWIN as there are difficulties with Microsoft C and "long long" variables

of less than 10KB to ensure scalability with the large number of clients anticipated. The BOINC system was set up to run each task at least three times and to validate only results for which two of them are identical. More recent versions of BOINC include a new feature called "Homogeneous Redundancy" [16] which provides a general solution for divergent applications like SixTrack by replicating tasks only on compatible hardware. Our solution allows us to use any Pentium or compatible hardware and to obtain identical results.

After successfully completing the initial tests, the general public were invited to sign up to the LHC@home project, SixTrack under BOINC, and over 30,000 people worldwide have subscribed up to 60,000 machines. Over the last year some two thirds of the 600,000 jobs for the LHC tune scan have been completed. This represents an estimated 300 CPU years, or 3 years dedicated usage of the NAP cluster. It should be emphasised that one part–time fellow is responsible for managing the work in reasonable batches of ten to twenty thousand jobs. The LHC@home subscribers, to whom we are extremely grateful, tend to be motivated by the allocation of BOINC credits, and may well lose motivation, during the periods of result analysis and the preparation of future runs.

While detailed statistics are not available at the present time, less than 3% of results are rejected by the BOINC quorum. This is comparable with the statistics available from other BOINC projects. We are confident that these results come from failing computers, network transmission problems, or other data corruption. Only BOINC validated results are returned to the end user.

## CONCLUSIONS

It is now possible for an Accelerator Physicist to make LHC Tracking studies with SixTrack on the CERN Batch System, on BOINC, on CPSS (and soon the GRID) by simply setting the runtime environment parameter "platform" to one of the above and making a cron table entry of a four line script to return results. A study may use all of the systems in parallel or on different subsets of the work to speed up completion of the study. This ten to hundredfold increase in computing capacity has been provided with an extremely modest hardware investment (a WWW PC server and backup for CPSS and two PC servers for BOINC) and a small manpower investment in the server management. The main cost is not so much in modifying the application but in testing and verifying the result reproducibility which is our primary goal. It is particularly important to applications running in a distributed environment in that it allows the utilisation of any available processor on which the application has been validated.

We believe that the methodology can be extended to other proprietary IEEE 754 compliant processors, such as the Apple Macintosh, SUN, or IBM Power PC. The incorporation of the necessary parentheses to ensure a unique order of evaluation in arithmetic assignments should provide identical results, at any level of optimisation, and with any Fortran 95 compliant compiler. The same technique could be applied to any C++ C99 standard compliant application.

## REFERENCES

[1] W. Herr, D. Kaltchev, E. McIntosh and F. Schmidt, "Large scale beam-beam simulations for the CERN LHC using distributed computing resources", EPAC'06, June 2006, Edinburgh.

[2] E. McIntosh and A. Wagner,"CERN Modular Physics Screensaver or Using Spare CPU Cycles of CERN's Desktop PCs", Computing in High Energy and Nuclear Physics 2004, Interlaken, p. 1055.

[3] F. Schmidt, "SixTrack – User Reference Manual", CERN SL/94-56, March 2000.

[4] J.L. Henning,"SPEC CPU2000: Measuring CPU performance in the new millennium", Computer, July 2000.

[5] M. Hayes, E. McIntosh and F. Schmidt, "The Influence of Computer Errors on Dynamic Aperture Results Using SixTrack", LHC-PROJECT-NOTE-309, CERN, January 2003.

[6] M. Giovannozzi and E. McIntosh, "Parameter scans and accuracy estimates of the dynamical aperture of the CERN LHC", EPAC'06, June 2006, Edinburgh.

[7] D. Goldberg, "What every computer scientist should know about floating-point arithmetic.", Computing Surveys, 1991, 23(1) p.5.

[8] D. Priest,"Differences among IEE 754 implementations", www.validlab.com/goldberg/addendum.html.

[9] T.C. Belding,"Numerical Replication of Computer Simulations: Some Pitfalls and How To Avoid Them", 2000, //www.citebase.org/abstract?id=oai:arXiv.org:nlin/0001057.

[10] V. Lefevre, J-M. Muller and A. Tisserand, "The Table Maker's Dilemma", IEEE Transactions on Computers, November 1998, Vol. 47 No. 11.

[11] V. Lefevre and J-M. Muller, "Worst Cases for Correct Rounding of the Elementary Functions in Double Precision", August 2003.

[12] P. Zimmermann, "MPFR: A Library for Multiprecision Floating-Point Arithmetic with Exact Rounding", 4th Conference on Real Numbers and Computers, 2000, Dagstuhl, p.89 (see also http://www.loria.fr/projets/mpfr/).

[13] "IBM Accurate Portable MathLib", http:/oss.software.ibm.com/mathlib/.

[14] F. de Dinechin, A. Ershov, and N. Gast, "Towards the post-ultimate libm.", IEEE 17th Symposium on Computer Arithmetic, June 2005, p. 288.

[15] D.P. Anderson, "BOINC: A system for public-resource computing and storage", IEEE/ACM International Workshop on Grid Computing (GRID'04), November 2004, p.4.

[16] M. Taufer, D. Anderson, P. Cicotti, and C.L. Brooks III, "Homogeneous Redundancy: a Technique to Ensure Integrity of Molecular Simulation Results Using Public Computing", 19th IEEE International Parallel and Distributed Processing Symposium (IPDPDS'05) Workshop, p.119a.