A Brief History of Computing leading to LHC@home
(from personal experience)

Eric McIntosh
cern.ch/mcintosh
eric.mcintosh@cern.ch
Accelerator chain at CERN, a complex business
I/O and NETWORKING

- Eyes, Scanner, Display/screen
- Ears, Microphone, Mouth, Speaker
- Disk, drum, SSD, Magnetic Tape, CD
- Punched Cards and Paper Tape
- Other special devices like HPD
- Just about anything, USB, DMA,....
- NETWORKING!!! The 1960's
- Connect to other computers and the world and the World Wide Web, WWW
Types of Computer

• Biological i.e. you and me “Wim” Klein
• Mechanical e.g. Abacus, Slide Rule, FACIT (still used in 1960)
• PUNCHED CARDS (Looms and IBM) 19th century already
• ELECTRO-MECHANICAL and ANALOG 20th century
• ELECTRONIC The first ENIAC IN 1937
• TRANSISTORISED 1960 the beginning of a revolution
A couple of mechanical calculators

An Abacus (works in 5s) in use for over 3000 years

And a FACIT as I used in 1958.
English Electric Deuce Computer 1960
Mercury Delay Lines
The Revolution

- Many electronic machines from the 1960s onwards
- IBM dominated, but CDC then Cray (the first supercomputer), English Electric, Siemens, Fujitsu, DEC, SUN, H-P, etc etc
- Best shown on the next slides with emphasis on Performance and Price
- The TRANSISTOR, CMOS/Silicon, ????
The EVOLUTION

- Mainframe and SuperComputer
- Minicomputer
- Workstation
- The PC
PARALLELISM

- Basically 2 main types
- Shared Memory or Networked
- OpenMP or MP Thread Parallel
- MPI (PVM) for Networked
- GPU Graphics Processing Unit
- Task Parallel e.g. SixTrack with little or no communication between tasks
- Pipelining/Vectorisation
Networked Computers

- IBM 7090 and 1401 (via magnetic tape)
- CDC 7600 and 6400 and 6500
- CRAY XMP and IBM, CDC, VAX Frontends and Tape Staging
- First centralised data acquisition – VAX to CRAY to IBM to Magnetic Tape
- SHIFT system at CERN
CERN Units

183 Dual 800MHz PIII 256K Cache Linux lxplus028 /2001
>1000 CERN Units per processor in 2019

20.5 FUJITSU M1800-20 UTS/M FOR77 5/91
20.3 IBM 9000/900 VM VSFORT OPT(3) XA mode 3/92
17.1 CRAY C90 (4.16ns) UNIX CFT77 vector 10/92
16.9 DEC 3000 AXP/400 (133MHz) VMS FTN T3.3 11/92
15.9 CRAY C90 (4.16ns) UNIX CFT77 scalar 10/92
15.3 IBM RISC 6000/970 50MHz -03 hssngl dyn 10/92
15.1 NEC SX-3 Super-UX f77sx vector 10/91
14.2 NEC SX-3 Super-UX f77sx 32bit vector 11/92
14.1 H-P 9000/735 99MHz f77 +03 -archive 11/92
13.5 H-P 9000/750 HP-UX f77 -0 dynamic 10/92
13.2 SGI R4000 50MHz Irix f77 -02 -mips2 dyn 10/92
12.3 AMDAHL 5990 VM/CMS VSFORT 2 7/88
11.8 CRAY Y-MP/864 UNIX CFT77 6.0 vector 10/92
11.5 NEC SX-2 SXOS VSFORT77SX vector 6/89
SixTrack Performance

- To be provided, Pre-processing, Tracking, Post-processing
- One turn, 10,000 steps, many loops over the (60) particles being tracked
LHC (Model) Summary

• 27KM circumference
• Magnets (Dipoles to 20-Pole), Cavities, Beam to Beam, Straight Sections, etc
• ~10,000 elements / steps of 50 types
• A bunch of 30 particle pairs (NOT 10**11)
• Initial conditions in phase space
  – Tune
  – Amplitude
  – Angle
Terminology

• A Study, typically a few thousand or more jobs from 2 to 10 hours CPU, $10^{**5/6}$ turns
• Needs LHC physical description, magnet errors, alignment errors.
• A Case (job) has one set of initial values
• Postprocessing is the amalgamation of all the results to define the Dynamic Aperture (from which 10% is subtracted).
The SixTrack Program

- 60,000 lines of standard Fortran 77
- Pre-processing, Tracking, Post-process
- Dimensioned for 60 particles, 30 pairs
- Memory requirement – 64 MegaBytes
- 500KB input – 10KB output (gzipped)
- It is NOT madX, replacing MAD 8/9
- Fortran is a Structured Programming Language, now using Fortran 2008
Computing at CERN

• Dominated by the needs of the experiments
• Accelerator design, a small fraction of the various mainframes (1964 – 1998) and the “PARC” IBM workstation cluster
• In 1997 the LHC Machine Advisory Committee recommended more tracking
• The “Numerical Accelerator Project”, NAP luck for me, F. Schmidt, and T. Pettersson
NAP Evolution

• A 10 processor Digital/Compaq Alpha TurboLaser (800 CERN Units)
• Added 10 Workstations (1,300 CUs)
• Overlapped by 20 DUAL 800Mz PIII’s (7,200 CUs)
• Today 64 Dual 2.4GHz PCs (51,200 CUs)
• Operated as “Fair Share” of the central Linux LSF Batch system lxbatch
The Idea (not original)

• Studies were still typically 1 tune, 60 seeds, up to 8 amplitudes, and 5 angles
• Use ~10000 Windows desktops at CERN to run SixTrack, a highly optimised LHC tracking program
• SixTrack was standard F77 and part of SPECFP2000 and today almost Fortran 2008
• Only 50KB (500KB) IN and < 2MB (6MB) OUT for ~ 1 to 10 hours CPU – ideal for networked computers
• At least double the tracking capacity and potentially provide an order of magnitude increase for zero financial investment
Initial Problems

• No compatible WINDOWS graphics – just dummied out the HBOOK calls, **not required**

• CR/LF in Windows – remove them on Linux when retrieving the result

• Lost particle processing 1000 times slower on Digital – check more often for NaNs and Infs
CPSS Project

• A. Wagner CERN/IT/WINDOWS provided a screen saver, Web Server and PERL interfaces for job submission and result retrieval

• SixTrack Checkpoint/Restart

• Transparent (almost) SixTrack run environment on Linux

• Worked well ..........until occasional RESULT differences
First real problem

• 1500 jobs, 60 seeds, 5 amplitudes, 5 angles, (v64lhc.D1-D2-MQonly-inj-no-skew) for 10,000 turns
• The final results, the minimum, average and maximum Dynamic Aperture were within 1% of the lxbatch results
• The average DA was within 3 parts in 1000
• Tried 600 seeds/15,000 jobs as final pre-production
• ……BUT…..
# Result Comparison

## LSF/Linux Results

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 1</td>
<td>11.27</td>
<td>12.20</td>
<td>13.17</td>
<td>15.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 2</td>
<td>12.18</td>
<td>13.69</td>
<td>15.46</td>
<td>30.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 3</td>
<td>13.90</td>
<td>14.83</td>
<td>16.14</td>
<td>45.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 4</td>
<td>16.29</td>
<td>17.32</td>
<td>18.08</td>
<td>60.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 5</td>
<td>15.50</td>
<td>16.30</td>
<td>17.34</td>
<td>75.00</td>
</tr>
</tbody>
</table>

## Windows CPSS Results

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 1</td>
<td>11.17</td>
<td>12.21</td>
<td>12.97</td>
<td>8.00 18.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 2</td>
<td>12.18</td>
<td>13.66</td>
<td>15.24</td>
<td>8.00 18.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 3</td>
<td>13.53</td>
<td>14.80</td>
<td>16.09</td>
<td>8.00 18.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 4</td>
<td>16.41</td>
<td>17.31</td>
<td>-18.00</td>
<td>8.00 18.00</td>
</tr>
<tr>
<td>v64lhc.D1-D2-MQonly-inj-no-skew5 5</td>
<td>15.60</td>
<td>16.30</td>
<td>17.15</td>
<td>8.00 18.00</td>
</tr>
</tbody>
</table>
One bit too many……..

- Careful checking of duplicate results, for one specific seed, identified a difference in the distance in phase space, between a particle pair, when computed on Windows 2000 and on Windows XP.

- Exhaustive (-ing) analysis identified one number 3.756403155274550e-09 was being input as HEX BE3022357D9B0651 on Windows 2000 as compared to HEX BE3022357D9B0650 on Windows XP (and on Linux)
…..but how often? how important?

• 600 fort.16 input files (Multipole Errors)
• 2364 blocks of 40 double-precision numbers
• 100,000 turns each involving 10,000 steps
• Quickly ran 2 times 600 jobs on W2000/XP
• 505 files affected (95 OK) with from 1 to 7 numbers being one bit too large
• Total of 1115 errors in 60 million numbers retest with F2003 input conversion
A known problem

- Depends on Compiler/OS
- Could be fixed by (over-)specifying the input values
- Decided to buy the LAHEY-FUJITSU If95 compiler for WINDOWS (already on Linux) to replace the obsolete COMPAQ compiler
- Surprisingly? Gave “IDENTICAL” results on Windows and Linux
Floating-Point Arithmetic

- Single Precision (SP)
  - 1, 8, 23 (32)

- Double Precision (DP)
  - 1, 11, 52 (64)

- Extended Precision (EP) A mongrel?
  - 1, 15, 64 (80)

- Quadruple Precision
  - 1, 15, 112 (128)

- Arbitrary Precision (Maple, MPFR, etc)
• 4 rounding modes
• We consider only “round to nearest” _rn
• Double Precision
• ~15 (and a bit) decimal digits
• Range from ~-10**308 to 10**308 but also NaNs and +/- Infinity
• ULP is Unit in the Last (binary) Place
IEE 754 (1985)

• Defines unique reproducible result for +, -, *, /, and sqrt – the correctly rounded result being the floating-point number closest to the exact result
• It is incomplete and open to interpretation
• Needs to be combined with the language standard
• Strict compliance conflicts with performance
• Does NOT cover Elementary Functions
• 60-bit word, 6-bit byte, big/little endian
HORRIBLE
Floating-Point issues
(Double Precision Extended)

• Extended (internal) 80-bit Precision EP
• (Double) rounding applied arbitrarily
• Fused Multiply Add
• SSE2 OK (but cannot use FMA in recent AVX extensions)
• DISABLE EP, in fact the default with lf95
• (“everything” else is disabled anyway)
EP Disabled

• Must NOT use libm,
• other libraries ????
• May introduce new problems in borderline evaluations
• Could affect performance (convergence)
• I contend that these cases need to be solved otherwise
• (Intel will make it the default! NEVER)
The beam-beam case

- While running some 400,000 2 hour jobs covering 1000 angles to prove CPSS
- Tried a study involving beam-beam interactions over a million turns
- Immediately detected a few result differences between INTEL IA32 and ATHLON AMD64 (also INTEL IA64)
- Traced back to an “exp” function - Not easy, but do-able with binary output
- Abandon the goal of reproducibility??? Abandon the whole idea!!!
Investigation

- Verified that IA64 was same as AMD64 (but see later)
- Found the log function similarly afflicted
- WWW search – insulted on a News Group
- Most problems/solutions eliminated because of the simple code generation
- Found several relevant libraries – MPFR, libultim IBM, libmcr SUN, 36 crlibm ENS
The libraries

- MPFR – arbitrary precision – slow
- libultim – 800 bits – too much/not enough
- libcmmr – arbitrary precision – slower
- crlibm – double precision – optimised and portable to any IEEE-754 compliant CPU
- Finally adopted CRLIBM from the Ecole Normale Superieur at Lyon
crlibm

- Delivers correctly rounded double precision results for the elementary functions
- Proven to do so
- Performance “comparable” to libm on average Testing now < 2%, not finished
- REQUIRES EP DISABLED
- Really more than I needed
Crlibm functions

- EXP, LOG, LOG10, SIN, COS, TAN
- ATAN, SINH, COSH
- ASIN, ACOS, now available
  - I wrote them and ATAN2 in terms of ATAN
  - NOT proven correctly rounded
- Each function has four rounding modes – nearest, up, down, to zero
- E.g. exp_rn, exp_ru, exp_rd and exp_rz
THE Solution

- Installed crlibm (portable for Linux and Windows with gcc and Lahey-Fujisu C)
- The numerical differences disappeared
- Performance was at worst 10% slower in the most difficult beam-beam case (but on portable code)
- The only subsequent numerical differences have been traced to failing computers (3 desktops and 1 lxbatch)
- The Intel microcode bug (2017)
Some simple test results

• ULP – One Unit in the Last Place of the mantissa of a floating-point number (one part in roughly 10**16)
  – libm/crlibm IA32: 304 differences of 1ULP
  – ibm IA32/IA64: 5 differences of 1ULP
  – libm IA32/AMD64: 7 differences of 1ULP
  – libm IA64/AMD64: 2 differences of 1ULP
  – libm/libm NO EP: 134623 differences of 1ULP

• NO differences with exp_rn

• 1,000,000 exp calls with random arguments (0,1)
...and with If95

- lahey/crlibm IA32: 134645 differences of 1ULP
- lahey IA32/IA64: 7 differences of 1ULP
- lahey IA32/AMD64: 7 differences of 1ULP
- lahey IA64/AMD64: 4 differences of 1ULP

• NO differences with exp_rn
## crlibm exp performance

<table>
<thead>
<tr>
<th></th>
<th>Average</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>libm</td>
<td>365</td>
<td>236</td>
<td>5528</td>
</tr>
<tr>
<td>crlibm</td>
<td>432</td>
<td>316</td>
<td>41484</td>
</tr>
<tr>
<td>libultim</td>
<td>210</td>
<td>44</td>
<td>3105632</td>
</tr>
<tr>
<td>mpfr</td>
<td>23299</td>
<td>14636</td>
<td>204736</td>
</tr>
</tbody>
</table>
When quadruple precision is not enough – The Table Maker’s Dilemma

• Rounding the approximation of f(x) is not always the same as rounding f(x)
• Worst case for exp(x), x=7.5417527749959590085206221e-10
• Binary example x=1. (52)1 *2-53
  exp(x)=1. (52)0 1 (104)1 010101…
• quad (112 bit) approximations :  
  1. (51)0 1 (60)0 and 1. (51)0 0 (60)1 are both within 1 Quad ULP but which rounded value is nearest?
BOINC

• The Berkeley Open Infrastructure for Network Computing (c.f SETI@home) was suggested by Dr Segal of the IT dep't

• Initial tests were very positive with 200,000 hosts reached very quickly in 2004

• LHC@HOME today – up to 500,000 computers, 1,800,000 CPUs/Threads, typically around 150,000 active tasks

• Beam-beam studies, 600,000 one million turn 10 hour jobs run successfully
BOINC ........

• Some 1,000,000 cases completed
• Every jobs is run twice (at least) and only identical results are accepted (NO EPSILON required)
• Estimate 3% of results are erroneous due to undetected hardware errors, overclocking, or transmission errors. These results are of course rejected (validation).
• Today, normally less than 1 in 10,000
Remote daemon status as of 13 Sep 2019, 13:04:51 UTC

Tasks by application

<table>
<thead>
<tr>
<th>Application</th>
<th>Unsent</th>
<th>In progress</th>
<th>Runtime of last 100 tasks in hours: average, min, max</th>
<th>Users in last 24 hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>SixTrack</td>
<td>515771</td>
<td>121060</td>
<td>2.3 (0.01 - 211.86)</td>
<td>3786</td>
</tr>
<tr>
<td>sixtracktest</td>
<td>138259</td>
<td>11237</td>
<td>1.3 (0.01 - 57.89)</td>
<td>234</td>
</tr>
<tr>
<td>CMS Simulation</td>
<td>200</td>
<td>1542</td>
<td>12.65 (0.62 - 18.5)</td>
<td>57</td>
</tr>
<tr>
<td>Theory Simulation</td>
<td>198</td>
<td>5568</td>
<td>16.68 (0.03 - 52.13)</td>
<td>110</td>
</tr>
<tr>
<td>ATLAS Simulation</td>
<td>3190</td>
<td>11670</td>
<td>20.39 (0.03 - 207.9)</td>
<td>313</td>
</tr>
<tr>
<td>Theory Native</td>
<td>18</td>
<td>1929</td>
<td>1.41 (0.01 - 26.52)</td>
<td>36</td>
</tr>
</tbody>
</table>

Upstream server release: 1.0.3
Database schema version: 27028
Task data as of 14 Sep 2019, 5:42:03 UTC
### 'Over' results

<table>
<thead>
<tr>
<th>State</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inactive</td>
<td>0</td>
</tr>
<tr>
<td>Unsent</td>
<td>479239</td>
</tr>
<tr>
<td>Unknown</td>
<td>0</td>
</tr>
<tr>
<td>In progress</td>
<td>119335</td>
</tr>
<tr>
<td>Over</td>
<td>1552230</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Outcome</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>0</td>
</tr>
<tr>
<td>Success</td>
<td>1469029</td>
</tr>
<tr>
<td>Couldn't send</td>
<td>0</td>
</tr>
<tr>
<td>Computation error</td>
<td>69578</td>
</tr>
<tr>
<td>No reply</td>
<td>8352</td>
</tr>
<tr>
<td>Didn't need</td>
<td>3460</td>
</tr>
<tr>
<td>Validate error</td>
<td>47</td>
</tr>
<tr>
<td>Abandoned</td>
<td>1764</td>
</tr>
</tbody>
</table>

### 'Success' results

<table>
<thead>
<tr>
<th>Validate state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>73273</td>
</tr>
<tr>
<td>Workunit error - check skipped</td>
<td>8</td>
</tr>
<tr>
<td>Checked, but no consensus yet</td>
<td>1691</td>
</tr>
<tr>
<td>Task was reported too late to validate</td>
<td>251</td>
</tr>
</tbody>
</table>

### 'Client error' results

<table>
<thead>
<tr>
<th>Client state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downloading</td>
<td>10103</td>
</tr>
<tr>
<td>Processing</td>
<td>0</td>
</tr>
<tr>
<td>Compute error</td>
<td>18512</td>
</tr>
<tr>
<td>Uploading</td>
<td>0</td>
</tr>
<tr>
<td>Done</td>
<td>861</td>
</tr>
<tr>
<td>Aborted by user</td>
<td>40300</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File Delete state</th>
<th># results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>242377</td>
</tr>
<tr>
<td>Ready to delete</td>
<td>0</td>
</tr>
<tr>
<td>Deleted</td>
<td>1226652</td>
</tr>
<tr>
<td>Delete Error</td>
<td>0</td>
</tr>
<tr>
<td>Total files deleted</td>
<td>1226652</td>
</tr>
</tbody>
</table>
The PARK, the FARM, the CLOUD

- Around one million registered hosts
- Almost 500,000 active/with credit
- Around 1,800,00 CPUs/Threads
- Around 400,000 Windows
- (About 164,000 claim to run Windows XP)
- 80,000 Linux
- 7,000 Darwin
- Around 100 ARM/Android
The Park, the Farm the Cloud

Around one million registered hosts
Almost 500,000 active/with credit
Around 1,800,000 CPUs/Threads
Around 400,000 Windows
  (About 164,000 claim to run Windows XP)
80,000 Linux
7,000 Darwin
Around 100 ARM/Android
• Am I obsessed about a numerical difference of 1ULP?
• It IS a problem for tracking studies, weather/climate prediction and other “chaotic” applications such as molecular systems
• Having eliminated ALL numeric differences SixTrack can be run on any IEEE 754 compatible hardware with identically replicated results (reportedly “probably impossible”)

51
Unfortunately, when it comes to floating-point arithmetic, the goal is virtually impossible to achieve. The authors of the IEEE standards knew that, and they didn't attempt to achieve it (i.e. Identical results).

As a result, despite nearly universal conformance to (most of) the IEEE 754 standard throughout the computer industry, programmers of portable software must continue to cope with
The next steps

- Extend to other C/C++ C99 compliant applications and compilers and GAMES? And Sixtracklib
- Already ported to Intel/AMD, Apple, ARM/ANDROID, Raspberry PI, IBM Power Series, GPUs, Linux, Windows, MacOS, PCs from Pentium 3 onwards
- ALWAYS MAINTAIN IDENTICAL DOUBLE PRECISION FLOATING-POINT RESULTS ...... 0 ULP DIFFERENCE