



2nd C++ Compiler for Run II Experiments ?



- CDF and D0 are using C++ as a primary programming language , depend on C++ compilers
- FORTRAN -> C++ transition degraded performance of the code
- CPU resources available are a limiting factor
- Performance of the code determines computing model
- both experiments would benefit from improving performance of the software
- Using single compiler: moved/moving from KAI (bought by Intel) to GCC
 - pros: less support effort
 - cons: single point of failure, CDF Computing Review'2004 pointed to it



2nd C++ Compiler for Run II Experiments ?



- CDF runs offline Production built in optimized mode
- Ratio "opt/non-opt" for *GCC* and *KCC* is very different (CDF):
 - *GCC*: "O3" improves speed by ~ x2 compared to non-opt
 - *KCC* : with O2 gain is ~ 30%
- Generation of MC datasets in 2004 took 3-4 months => with *GCC* forced to run optimized MC code as well
 - Monte Carlo code was failing in optimized mode with *GCC* 3.1, primary suspect - FORTRAN compiler bug (*GEANT3!*)
- CDF was forced to try the latest version of *GCC* (3.4)
 - First results encouraging, validation in progress
- Still *GCC* is a single point of failure - what about other compilers?



INTEL compilers



- C++ Working Group: Intel C++/FORTRAN is the first choice
- Intel C++ on Intel CPU's improves speed by ~50% compared to GCC
- What if it is 20-30% for AMD processors?
 - 20-30% reduction in the number of CPU's needed
 - or 20-30% decrease in the processing time
- INTEL doesn't have debugger, claim to be compatible with GGB
- INTEL has performance analyser !
- Tested in the field: ROOT development team uses INTEL compiler
- evaluation of the INTEL C++ seems to be the right step
- CDF would be willing to port its software to INTEL compiler and do the evaluation
 - relatively small task: Rob Kennedy did INTEL port once