AN OpenMP PARALLELIZATION OF REAL-TIME PROCESSING OF CERN LHC BEAM POSITION MONITOR DATA

H.Renshall, L.Deniau, CERN, Geneva, Switzerland

Abstract

SUSSIX is a FORTRAN program for the post processing of turn-by-turn Beam Position Monitor (BPM) data, which computes the frequency, amplitude, and phase of tunes and resonant lines to a high degree of precision. For analysis of LHC BPM data a specific version run through a C steering code has been implemented in the CERN Control Centre to run on a server under the Linux operating system but became a real time computational bottleneck preventing truly on-line study of the BPM data. Timing studies showed that the independent processing of each BPMs data was a candidate for parallelization and the Open Multiprocessing (OpenMP) package with its simple insertion of compiler directives was tried. It proved to be easy to learn and use, problem free and efficient in this case reaching a factor of ten reductions in real-time over twelve cores on a dedicated server. This paper reviews the problem, shows the critical code fragments with their OpenMP directives and the results obtained.

THE PROBLEM

SUSSIX is a FORTRAN program for the post processing of turn-by-turn Beam Position Monitor (BPM) data, which computes the frequency, amplitude, and phase of tunes and resonant lines to a high degree of precision through the use of an interpolated Fast Fourier Transform (FFT). Analysis of such data represents a vital component of many linear and non-linear dynamics measurements.

For analysis of LHC BPM data a specific version *sussix4drive*, run through the C steering code *Drive God Lin*, has been implemented in the CERN Control Centre (CCC) by the beta-beating team. Analysis of all LHC BPMs, however, represents a major real time computational bottleneck in the control room, which has prevented truly on-line study of the BPM data. In response to this limitation an effort has been underway to decrease the real computational time, with a factor of 10 as the target, of the C and FORTRAN codes by parallelising them.

SOLUTIONS CONSIDERED

Since the application is run on dedicated servers in the CCC the obvious technique is to profit from the current multi-core hardware: 24 cores are now typical. The initial thinking was to parallelise the FFT code.

The first attempts were to try a parallelised FFT from the Numerical Algorithms Group (NAG) `fsl6i2dcl` library for SMP and multicore processors together with the Intel 64-bit FORTRAN compiler and the Intel maths kernel library recommended by NAG. This library uses the OpenMP technology.

Various NAG examples of enhanced routines were run (but not the multi-dimensional FFTs) and all slowed down in real time using more cores. This was not surprising since the examples only take milliseconds, comparable to the overhead to launch a new thread. Also it was found that the SUSSIX application calls `cfft` (D704 in the CERN program library), which maps onto NAG `c06ecf`, which had not yet been enhanced. This led to making a detailed central processing unit (CPU) profiling of the application.

Profiling the application (with gprof) showed that in fact only 7.5% of the CPU time was spent in `cfft` while 70% was spent in a function `zfunr` searching for the maximum of the Fourier spectra with large numbers of executions of an efficient inner loop of BPM data over many turns. This loop could not be improved (`maxd` turns of a BPM data is typically 1000):

```
double complex zp,zpp,zv
zpp=zp(maxd)
do np=maxd-1,1, -1
  zpp=zpp*zv+zp(np)
enddo
```

It was decided to try and parallelise SUSSIX directly using the OpenMP implementation supported by the Intel and GCC compilers. The home web site is www.openmp.org and an excellent tutorial at computing.llnl.gov/tutorials/openMP was the main reference. Examination of the code granularity revealed that the highest level of independent code execution was over the processing of individual BPM data.

The pure FORTRAN offline version was parallelised first by adding OpenMP parallelisation directives around the main BPM loop. In this version each BPMs data is in a separate physical file hence they could be opened and read in parallel:

```
!$OMP PARALLEL DO PRIVATE(n,iunit,
                          filename,nturn)
!$OMP& SHARED (isix,ntot,narm,iana,…,
               iicf)
do n=1,ntot ! Parallel loop over BPM
  call datspe(iunit,idam,ir,nt1,nt2,
             nturn,…,iana)
  call ordres(eps,narm,nrc,idam,n,nturn)
enddo
!$OMP END PARALLEL DO
```


ISBN 978-3-95450-116-8

In addition, `!$OMP THREADPRIVATE` directives were added for all non-shareable variables in the called subroutine trees of `datspe` and `ordres`.

This gave good scaling up to 10 cores on a public shared 16-core CERN computer centre *lxplus* machine so was worth extending to the target mixed C and FORTRAN version to be run in the control room. In this version the BPM data are all read into memory from a single file before being passed sequentially into the processing loop. The FORTRAN BPM processing loop is called from C and was parallelised with the similar OpenMP C-code syntax and gave the same scaling result:

```
#pragma omp parallel private(i,ii,ij,kk)
#pragma omp for
for (i=pickstart; i<=maxcounthv; i++) {
  sussix4drivenoise_(&doubleToSend[0],
                     &tune[0],&amplitude[0])
#pragma omp critical
// … I/O loop with sequential execution
}
```

The FORTRAN `datspe` and `ordres` call trees were unchanged.

The OpenMP directives multi-thread the code and the threads then map onto physical cores in a multi-core machine. The run-time environment variable `OMP_NUM_THREADS` instructs OpenMP how many threads, hence cores, it can use for an execution and enables easy measurement of the scaling.

Since the order of processing of individual BPMs is arbitrary the results file is post-processed by the Unix sort called as part of the application to give the same results output as a non-parallel execution.

A test case of real 1000 turns LHC BPM data, analyzed to find 160 lines, was run on a reserved 24 cores machine *cs-ccr-spareb7* in the LHC CCC.

A normal run of this test case takes about 50 seconds on this machine. The observed wall-time speedup of C-FORTRAN SUSSIX as a function of the number of cores (from E. Maclean,CERN,BE) is shown below in figure 1.

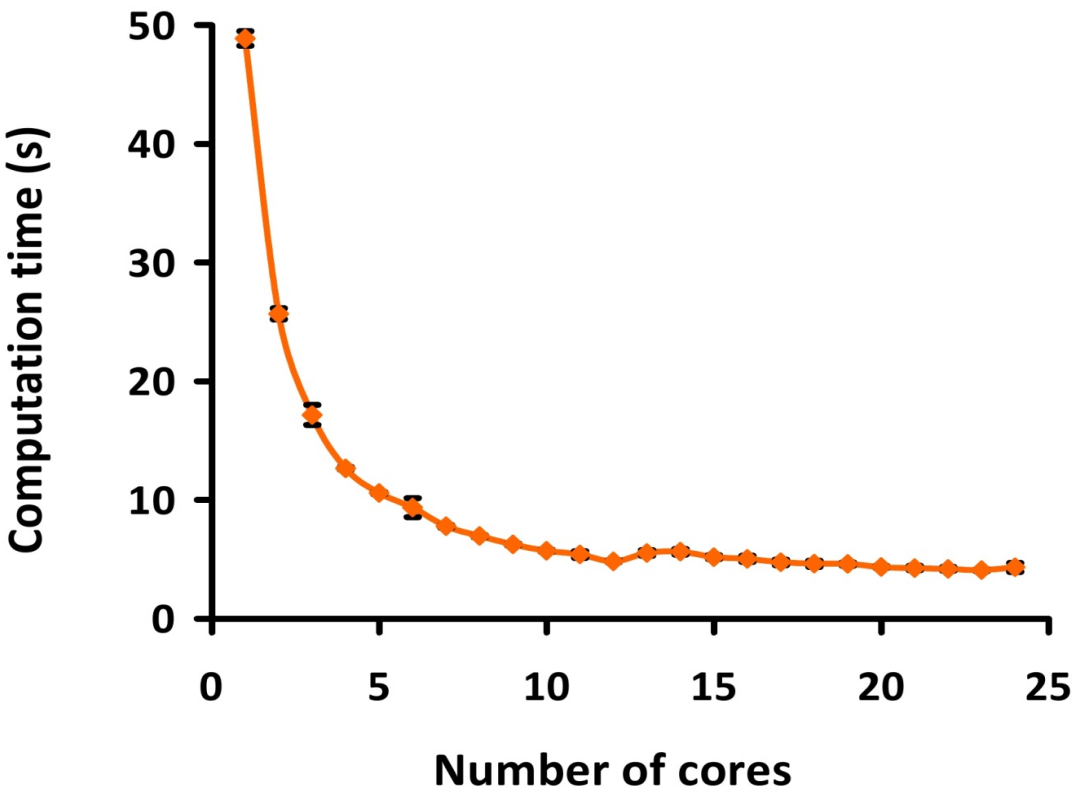

Figure 1: Real time in seconds to solution with increasing number of cores.

CONCLUSIONS

About a factor of 10 improvement in the real computation time has been realized for this test case saturating at 12 cores, probably due to memory bandwidth limits (other possible causes were investigated). For the study of amplitude detuning reported in CERN-ATS-Note-2011-052 MD [1] the parallelised C-FORTRAN SUSSIX was utilized within the LHC beta-beat graphical

ISBN 978-3-95450-116-8

user interface [2] and the target tenfold real-time reduction was verified in practice.

There was a modest learning curve for OpenMP but the Lawrence Livermore National Laboratory tutorial was easy to read and understand with its simple FORTRAN and C code fragment examples. Most of the code changes were insertion of OpenMP directives and these were correctly implemented by the Intel FORTRAN and GCC compilers.

This simple robust technique could be of interest to other real-time applications where a significant fraction of the CPU time is spent in independent calculations which have been implicitly serialized by being written in standard FORTRAN or C.

REFERENCES

[1] M. Albert et al., "Non-linear beam dynamics tests in the LHC" CERN-ATS-Note-2011-052 MD, July 2011.

[2] M. Aiba et al., "Software package for optics measurement and correction in the LHC", CERN-ATS-2010-092.

ISBN 978-3-95450-116-8