Efficient Libraries Use

Dr. Dimitris Dellis

GRNET

Athens, 12 Dec. 2018

Libraries in HPC Enviroment I

- We have our code, we are happy, it is optimized.
- Is this enough ?
- A very common and small piece of code : matrix - matrix multiplication, essentially one line of code inside three loops.
- Other similar procedures : Matrix Vector multiplication.
- Both appear in many physics transformations (Like rotation of rigid bodies, correlation functions etc)
- In Fortran, suppose square arrays (N=M=K) for simplicity

Libraries in HPC Enviroment II

- We are using a Broadwell architecture CPU. This means AVX2 and FMA instructions are available.

```
do i = 1, NMAX
   do j = 1, NMAX
         c(i,j)= ZERO
         do ij = 1, NMAX
           c(i,j)=c(i,j)+a(i,ij)*b(ij,j)
         enddo
   enddo
enddo
```

Libraries in HPC Enviroment I

- No apparent code optimizations.
- Any suggestion ?
- If not, we have to live with this piece of code.

Libraries in HPC Enviroment:Compilers and Flags I

- Some Equations :

$$A \cdot B = C$$

$$\begin{bmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ .. & .. & \dots & .... \\ A_{M1} & A_{M2} & \dots & A_{MN} \end{bmatrix} x \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1K} \\ B_{21} & B_{22} & \dots & B_{2K} \\ \dots & \dots & \dots & \dots \\ B_{N1} & B_{N2} & \dots & B_{NK} \end{bmatrix} = \begin{bmatrix} C_{11} & C_{12} & \dots & C_{1K} \\ C_{21} & C_{22} & \dots & C_{2K} \\ \dots & \dots & \dots & \dots \\ C_{M1} & C_{M2} & \dots & C_{MK} \end{bmatrix}$$

  - For each C element N multiplications and N additions are required
  - Total $N \times M \times K$ multiplications and $N \times M \times K$ additions, $=2 \times N \times M \times K$ Floating point operations.
  - If $M = N = K = 1000$, 2×10^9 floating point operations are required. If a machine can perform these operations in 1 sec, its performance is 2 GFLops

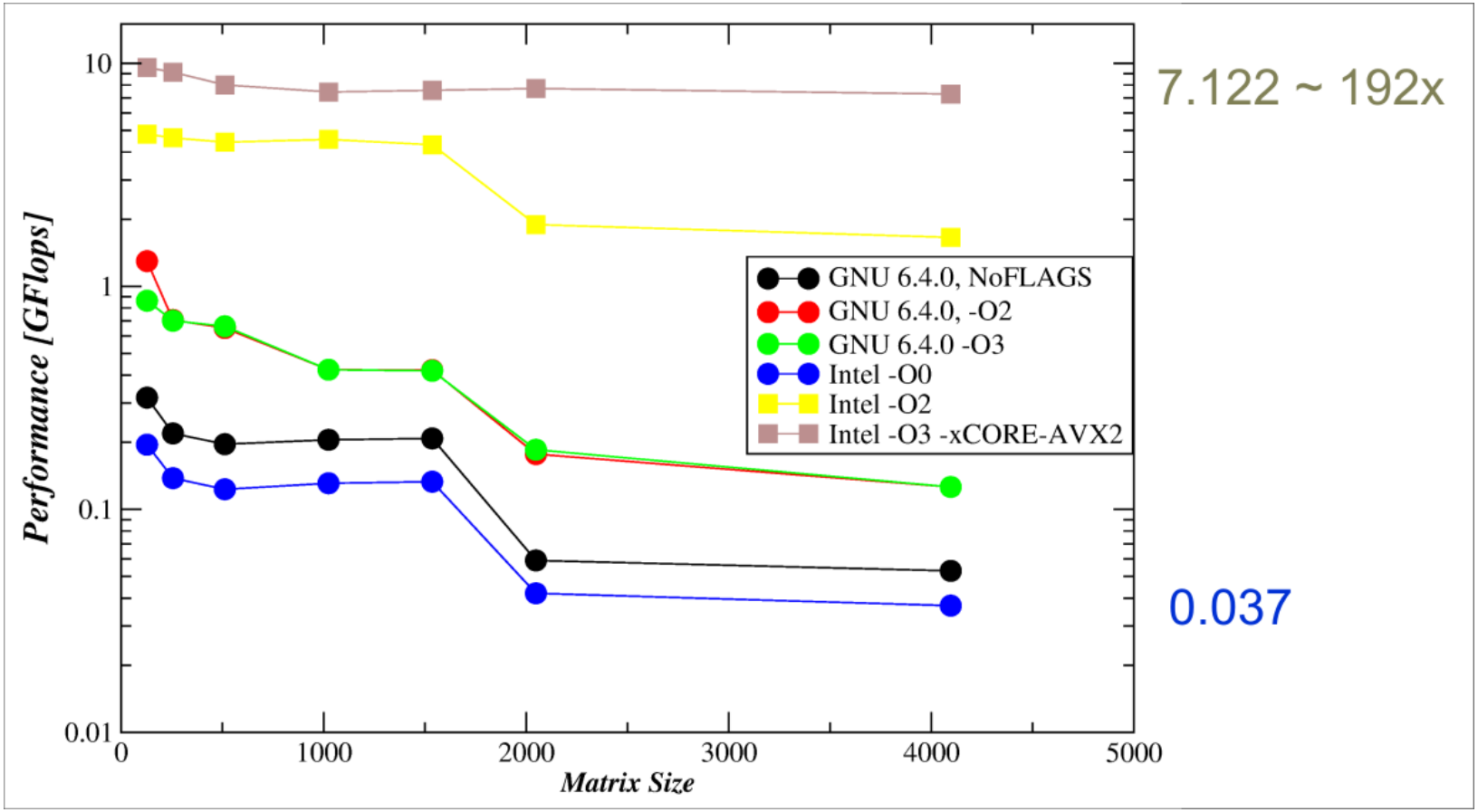
Libraries in HPC Enviroment:Compilers and Flags I

- Let's try recent versions of GNU and Intel Compilers (6.4.0 and 17.0.5 respectivelly), playing with compiler flags :
- gfortran code.f -o code.x
- gfortran -O2 code.f -o code.x
- gfortran -O3 -march=broadwell -mtune=broadwell -mavx2 -mfma code.f -o code.x
- ifort -O0 code.f -o code.x
- ifort -O2 code.f -o code.x
- ifort -O3 -xCORE-AVX2 code.f -o code.x

Libraries in HPC Enviroment:Compilers and Flags II

- In fact, examples are more compilcated, includind C code, for memory allocation and timing, conditional defines through Makefile etc. Look at the Makefile for details.
- Look in Performance results. The example codes report the performance as function of arrays size, in order to have wider view of performance (and its behavior as function of problem size).
- Speed Units are GFlops.
- Lets discuss the next performance data.

MxM Performance : Compilers and Flags



Libraries in HPC Enviroment:Compilers and Flags I

- It is clear that Intel compilers at high optimization levels are the winner.
- Although it is the slower at level 0.
- It seems that Intel 17 without optimization flags, corresponds to -O2.
- Results comparison, yields a speed up of 200 $\times$.
- We are happy with this 200$\times$ speed up ?
- Not for me. Go ahead with other possibilities.

Libraries in HPC Enviroment:Compilers and Flags II

- Let's try some libraries that implement this matrix-matrix multiplication operation.
- Code should change. Instead of the triple loop one has to call a library function with arrays and sizes as arguments.
- Commonly accepted standard is the BLAS/LAPACK notation :

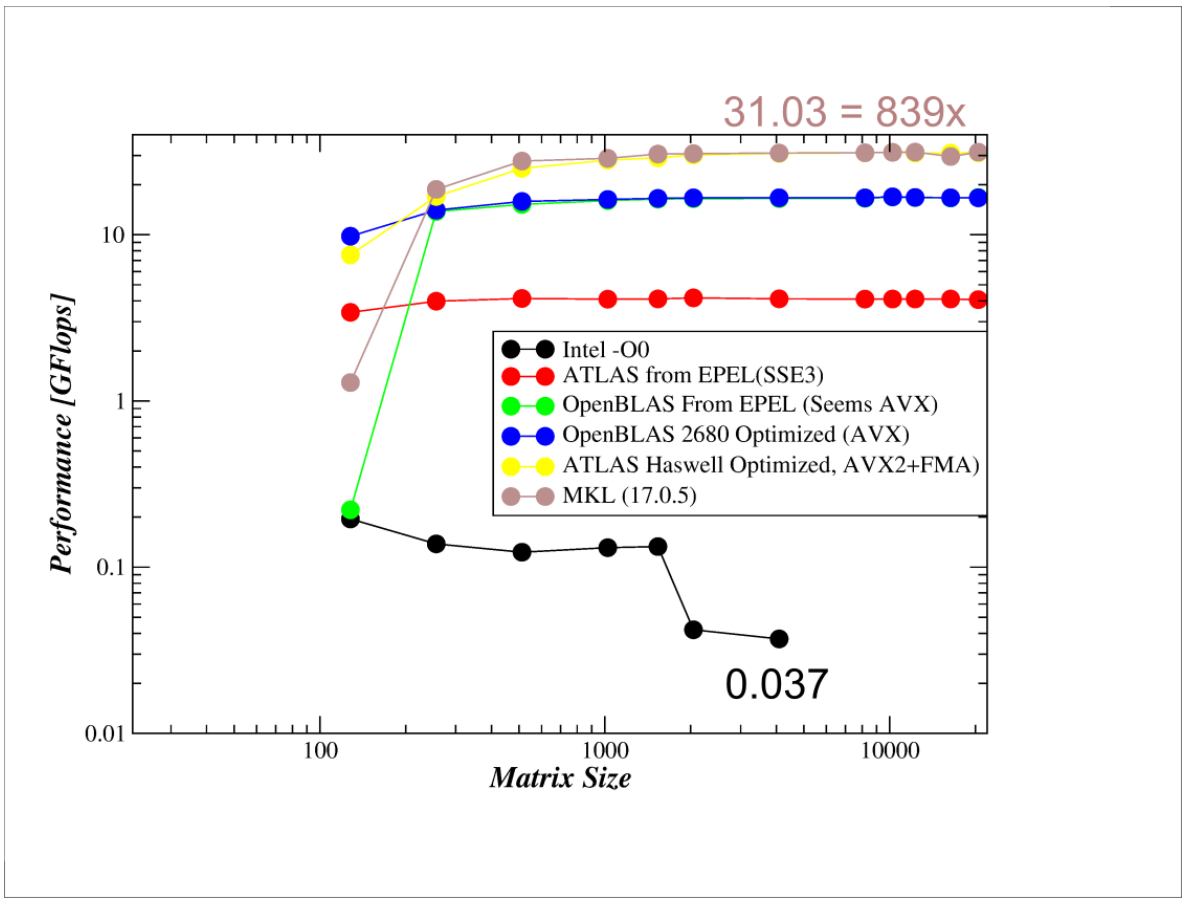
```
call dgemm('N','N',NMAX,NMAX,NMAX,ONE,a,lda,b,ldb,ONE,c,ldc)
```

- For not obvious vars meaning, look in BLAS/Lapack Manuals.

Libraries in HPC Enviroment:Compilers and Flags III

- Tested implemetations are : MKL, natively compiled OpenBLAS 0.2.20, ATLAS 3.11.39, OS/EPEL repositories implementations of OpenBLAS 0.2.20 and ATLAS. More possibilities could be presented.
- Let’s look in obtained Performance results as function of array size and BLAS Implementation.

MxM Performance : Code vs Libraries



Libraries in HPC Enviroment:Compilers and Flags I

- Results, look interesting : At least 50 times faster at low i.e. ≤ 128 sizes, up to 839 times faster at size 4096, slightly higher at higher N..
- Are you happy with this speedup ?
- What happens at low N, that are very common in codes (like 3x3 for obvious reasons) ?
- At low sizes we see variation in performance betwwen implementations.

Libraries in HPC Enviroment:Compilers and Flags II

- libSMM (Small Matrices Multiplication) developed by Cray Inc. targeting mainly *ab-initio* codes, where multiplications of small *exotic* (whatevet it means) sizes is frequent.
- Let's compare LibSMM to Intel MKL for array multiplications of sizes up to 32.
- libSMM is a wrapper that either runs its own unrolled code) or calls the underlying BLAS (MKL in our case).
- Note : libSMM compilation takes almost one day on a single 20cores node...
- Again we have to alter the code : Instead of

Libraries in HPC Enviroment:Compilers and Flags III

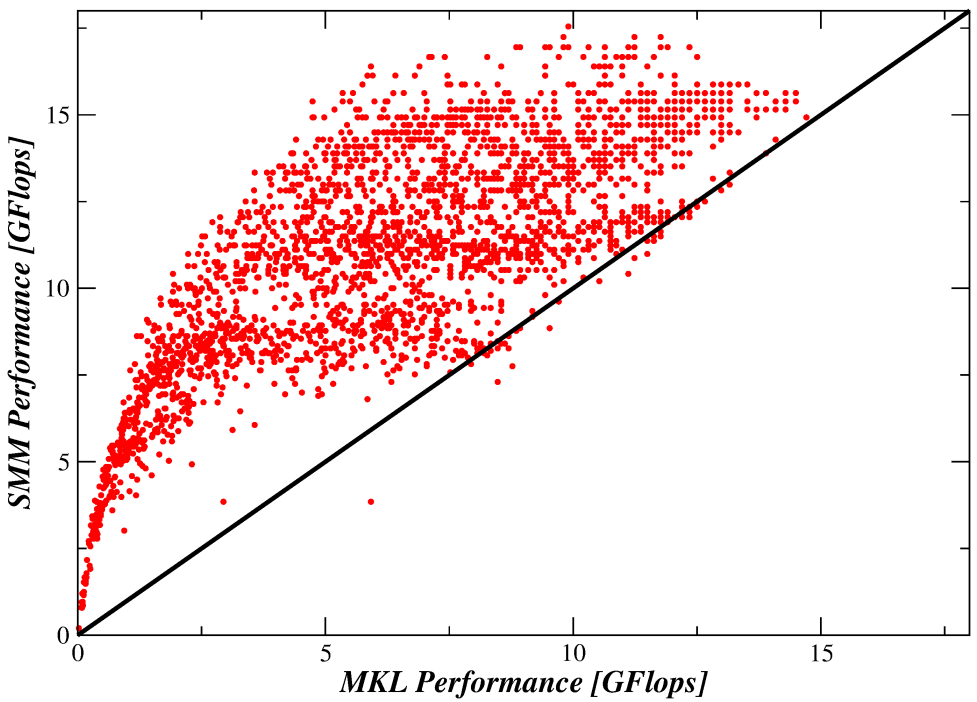
```
call dgemm('N','N',NMAX,NMAX,NMAX,ONE,a,lda,b,ldb,ONE,c,ldc)
```

- We have to modify it to look (for double precision) like:

```
call smm_dnn(M,N,K,A,B,C)
```

- In next slide, the performance ratio of libSMM over MKL at the same size is compared for all combinations of dimension between 1..32.
- All results for single core...

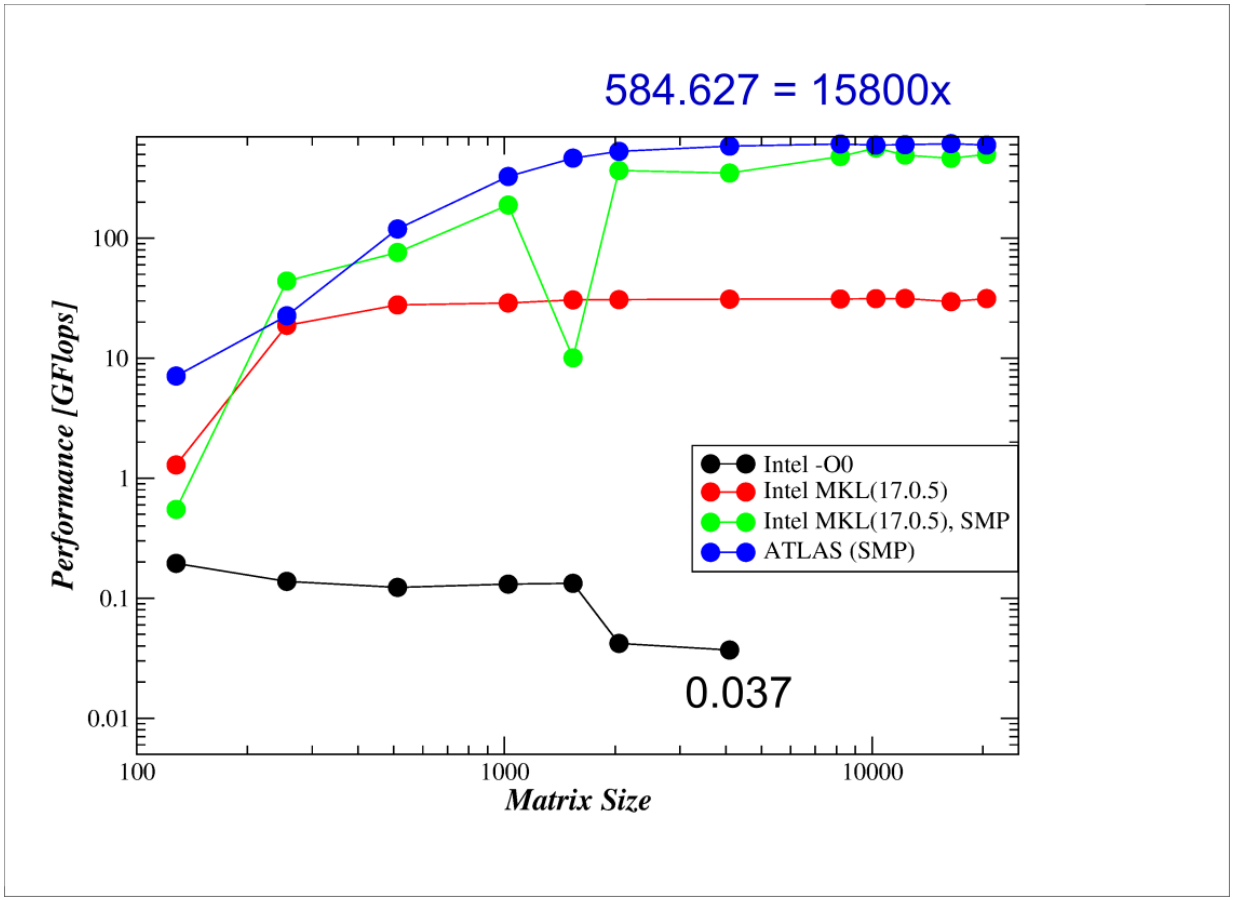
MxM Performance : Extreme Scaling, LibSMM



Libraries in HPC Enviroment:Compilers and Flags I

- On the average a 2 times speed up with respect MKL at low (up to 32 array sizes),
- Are you happy whith this ?
- Not me, if my code uses say 75% of time in this type of operations.
- Let’s check SMP implementations.

MxM Performance : Code vs Libraries and SMP



Libraries in HPC Enviroment:Compilers and Flags I

- Things are more interesting. We could use more than one core for our job.
- Speed up up to 15800 $\times$ is not negligible.
- Especially if it is 31600$\times$ on fat nodes (with their 40 cores)
- Use of GPUs implies (for well written codes) an additional speedup of $\sim$ 2.6x.
- A speedup of 15800x looks much better, if it is combined with a $\sim$ 2.6x of GPU speedup.

Libraries in HPC Enviroment:Concluding I

- An (essentially) single line of code, could singificantly speed up an application.
- A recent application profile (see next session for profiling) shows that more than half of execution time is consumed in this type of operations that potenially could be negligible.
- Role of profiling : In previous PRACE training @GRNET, a user realized that the procudure that (s)he was thinking that takes most of time, at the end was not the bottleneck after application profiling.

Libraries in HPC Enviroment:Concluding II

- On ARIS and most (if not all) PRACE systems, all these speedup techniques are taken into account in compilation of end users applications.
- Usually, this type of optimizations are not available to small laboratory clusters, where the *install it from a repository* is the dominant way of software installation method.

Libraries in HPC Enviroment:Concluding I

- Other Libraries that could exhibit similar (or even better) performance behavior are : FFTW, Other Linear Algebra Packages like ELPA, I/O related libraries like SionLib for I/O etc.

Questions ?