IBM ESSL – Engineering and Scientific Subroutine Library

ESSL includes BLAS routines, LAPACK routines, FFTs, and much more.

Three main libraries :

(1) -lessl single-threaded routines for POWER8 CPUs)

(2) -lesslsmp multi-threaded routines for POWER8 CPUs, uses IBM XL OpenMP runtime

(3) -lesslsmpcuda offload to GPUs for a subset of routines including Level 3 BLAS (dgemm etc.)

Documentation … get the ESSL pdf file :

https://www.ibm.com/support/knowledgecenter/SSFHY8/essl_content.html

On summitdev, the ESSL module is loaded by default when you log in.

There is support for 64-bit integer arguments with -lessl6464 -lesslsmp6464.

The ESSL SMP CUDA library has Level3 BLAS routines. All matrices use host memory, but data can be pinned, The default is to use up to all of the devices on a node, depending on matrix dimensions.
You can control the choice of GPU devices with env variable CUDA_VISIBLE_DEVICES.
The default execution mode is hybrid …a routine like zgemm will use CPUs and GPUs.

Control of threading is via the OMP_NUM_THREADS environment variable.

export ESSL_CUDA_PIN=no , yes, or pinned (default = no; pinned means the data is already pinned)
export ESSL_CUDA_HYBRID=no, yes

Examples of compilation/linking with ESSL on summitdev

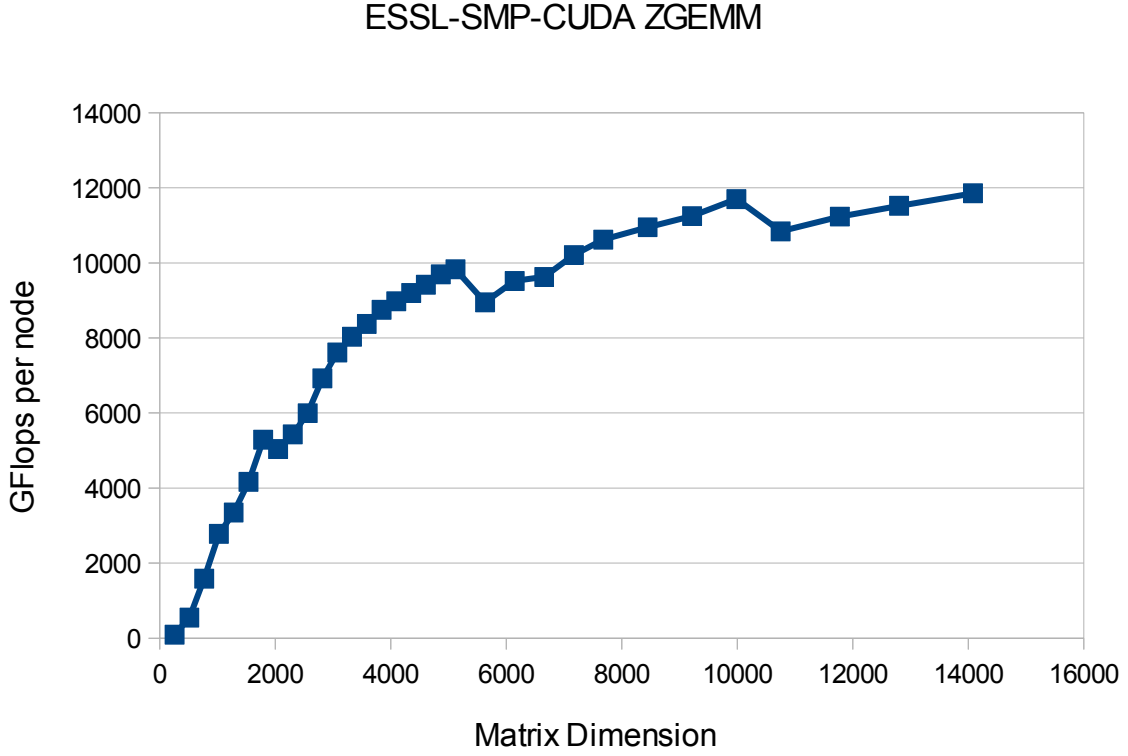
```
[summitdev-login1:~/codes/flops] module list

Currently Loaded Modules:
  1) xl/20161123                       4) xalt/0.7.5
  2) spectrum_mpi/10.1.0.2-20161130    5) DefApps
  3) hsi/5.0.2.p5                      6) essl/5.5.0-20161110

[summitdev-login1:~/codes/flops] make flops
mpicc -g -O2 flops.c -o flops -lessl
/usr/bin/ld: cannot find -lessl    … need to add -L${OLCF_ESSL_ROOT}/lib64

[summitdev-login1:~/codes/flops] echo $LD_LIBRARY_PATH (or env | grep OLCF)
/sw/summitdev/essl/5.5.0-20161110/lib64:...

[summitdev-login1:~/codes/flops] cat makefile
ESSL = -L${OLCF_ESSL_ROOT}/lib64 -lessl
...

[summitdev-login1:~/codes/flops] make flops
mpicc -g -O2 flops.c -o flops -L/sw/summitdev/essl/5.5.0-20161110/lib64 -lessl
```

With PGI compilers or clang, linking will fail with unresolved symbols from IBM XL Fortran.
Check : readelf -d ${OLCF_ESSL_ROOT}/lib64/libessl.so
=> ESSL library needs libxlf90_r.so.1 and libxlfmath.so.1 … IBM XL Fortran libraries

Solution : export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:${OLCF_XL_ROOT}/lib

With this addition to LD_LIBRARY_PATH, you can link ESSL using PGI or clang compilers.

To do for summitdev : adjust the modules infrastructure to include the path to the XL Fortran libraries, whenever the ESSL module is loaded.



Measurement on summitdev using one node, 4 MPI ranks, 1 rank per GPU.
export ESSL_CUDA_HYBRID=no; using default pinning = not pinned

Getting ~12 TFlops for large enough matrices.
GPU offload is effective for matrix dimensions > 512 or so.
CPU performance tops out at ~520 GFlops per node.

For more performance : pin matrix data and export ESSL_CUDA_PIN=pinned

For similar plug-in BLAS offload, check out NVBLAS.

FFTW3 FFTs on POWER8 CPUs with ESSL

FFTW3 version 3.3.5 has built-in support for POWER8 SIMD instructions.
IBM ESSL includes FFTW3 interfaces : libfftw3_essl.a (link with -lfftw3_essl).
The ESSL routines provide better performance for large FFT dimensions.

ndim	ORNL MFlops	build MFlops	ESSL-lib MFlops
16	7407.2	7121.0	1403.7
32	6693.1	11434.3	2726.5
64	7849.8	11815.9	4482.3
128	8893.6	11895.7	8057.3
256	8054.5	10412.4	11799.8
512	7681.7	10654.9	13593.5
1024	8211.0	11099.7	13353.3
2048	7670.7	10869.7	14872.0
4096	7063.1	10740.6	14039.0
8192	7042.3	8940.7	11927.8

1d complex FFTs with doubles

In the table: ORNL = library from “module add fftw/3.3.5” on summitdev
build = fftw-3.3.5 built from source
ESSL-lib = fftw3 routines from libfftw3_essl.a

To do : get a better build of fftw-3.3.5 on summitdev.
Configuration script for including support for the GNU OpenMP runtime :

```
#!/bin/bash
export CC=gcc; export F77=gfortran
export OMPI_CC=gcc; export MPICC=mpicc; export PTHREAD_CC=gcc
 ./configure --prefix=$HOME/fftw-3.3.5 --enable-mpi --enable-fma --enable-vsx  --enable-openmp
```