1. The compiler does 2 jobs: compile and link. Google commands like -o and -c. Read what you did in keyALGs::sourceCppExtra, and the template in tryCuda.

2. system("C:/rtools40/mingw64/bin/g++.exe -std=gnu++11 -Ofast -Wall -mfpmath=sse -msse2 -mstackrealign test.cpp -shared -o test.dll") is to build a shared library. Change *.dll* to *.exe* for executable.

*mfpmath=sse -msse2* is to enable SIMD for math operations. You may want to add *-march=native* if you have a more advanced CPU. mstackrealign is also good for performance.

system("C:/rtools40/mingw64/bin/g++.exe -std=gnu++11 -Ofast -Wall -mfpmath=sse -msse2 -mstackrealign -c test.cpp -o test.o") is to build an object file, which contains assembly codes.

system('"C:/Program Files (x86)/Microsoft Visual Studio/2019/Community/VC/Tools/MSVC/14.29.30133/bin/Hostx64/x64/dumpbin.exe" /EXPORTS test.dll') to see what is inside the dll. It seems GCC's nm.exe and dumpbin do not work on dlls!

#ifdef \_\_cplusplus    
extern "C" {  
#endif  
  // \_\_declspec(dllexport) // uncomment this line for MSVS.  
  void test(double \*x, double \*y, int \*size, double \*z);  
    
#ifdef \_\_cplusplus    
}  
#endif  
   
void test(double \*x, double \*y, int \*size, double \*z)  
{  
  for(int i = 0; i < \*size; ++i)  
  {  
    z[i] = x[i] \* y[i];  
  }  
}

3. Using GCC to compile with a dll, do something like this:

system('C:/rtools42/x86\_64-w64-mingw32.static.posix/bin/g++.exe -std=gnu++17 -O3 -Wall -mfpmath=sse -msse2 -mstackrealign -o test.exe src/tryXgboostCapi.cpp -LC:/Users/i56087/Documents/R-4.2.0/library/xgboost/libs/x64 -lxgboost')

Notice you have to first put -L with specify the dll's directory, and then -l which links to the dll itself without the file name extension. The functionality has been implemented in keyALGs::sourceExtraCpp(): e.g. sourceCppExtra(  
  "src/tryXgboostCapi.cpp", verbose = 1, # cacheDir = "./",  
  dllsNeeded = "C:/Users/i56087/Documents/R-4.2.0/library/xgboost/libs/x64/xgboost.dll",  
  rebuild = 1)

Running .exe or .dll should have copied those linked dlls in the same directory, OR, dlls have been loaded into the R environment using dyn.load()!

Notice tryXgboostCapi.cpp should include the .h file as extern C code.

4. In CUDA, a grid consists of blocks that are independent. A block consists of warps of threads. There are 32 threads per **warp**. A single instruction is issued simultaneously to all threads in a **warp**. If an operand is not ready then **warp** will stall. Registers and shared memory are allocated for a **block** as long as that **block** is active. Once a **block** is active it will stay active until all threads in the **block** have completed.

5. In Linux for dynamic linking to .so file like "libprofiler.so", remove the prefix "lib", e.g., -L"/directory/path" -l"profiler".

6. For installed Linux packages, the executable and dynamic linking libraries are most likely reside in subdirectories in /usr. Use   find /usr  -name "libprofiler.so"    to search for the files.

7. Compile C++ using Windows command prompt: the first step is to reset the environment variables like:

set PATH=’C:/rtools42/x86\_64-w64-mingw32.static.posix/bin’;%PATH%

Compare the difference between PATH in a bare Windows command prompt and PATH in Rstudio’s terminal, and you will see the difference.