Starting with GROMACS and OpenCL

Now that GROMACS has been ported to OpenCL, we would like you to help us to make it better. Why? It is very important we get more projects ported to OpenCL, to get more critical mass. If we only used our spare resources, we can port one project per year. So the deal is, that we do the heavy lifting and with your help get all the last issues covered. Understand we did the port using our own resources, as everybody was waiting for others to take a big step forward.
The below steps will take no more than 30 minutes.]

Getting the sources

All sources are available on Github (our working branch, bases on GROMACS 5.0). If you want to help, checkout via git (on the command-line, via Visual Studio (included in 2013, 2010 and 2012 via git_plugin), Eclipse or your preferred IDE. Else you can simply download the zip-file. Note there is also a wiki, where most of this text came from. Especially check the "known limitations".
To checkout via git, use:
git clone git@github.com:StreamHPC/gromacs.git

Building

You need a fully working building environment (GCC, Visual Studio), and an OpenCL SDK installed. You also need FFTW. Gromacs installer can build it for you, but it is also in Linux repositories, or can be downloaded here for Windows. Below is for Linux, without your own FFTW installed (read on for more options and explanation):
mkdir build
cd build
cmake .. -DGMX_BUILD_OWN_FFTW=ON -DGMX_GPU=ON -DGMX_USE_OPENCL=ON
-DCMAKE_BUILD_TYPE=Release

There are several other options, to build. You don't need them, but it gives an idea what is possible:

-DCMAKE_C_COMPILER=xxx equal to the name of the C99 compiler you wish to use (or the environment variable CC)
-DCMAKE_CXX_COMPILER=xxx equal to the name of the C++98 compiler you wish to use (or the environment variable CXX)
-DGMX_MPI=on to build using an MPI wrapper compiler. Needed for multi-GPU.
-DGMX_SIMD=xxx to specify the level of SIMD support of the node on which mdrun will run
-DGMX_BUILD_MDRUN_ONLY=on to build only the mdrun binary, e.g. for compute cluster back-end nodes
-DGMX_DOUBLE=on to run GROMACS in double precision (slower, and not normally useful)
-DCMAKE_PREFIX_PATH=xxx to add a non-standard location for CMake to search for libraries
-DCMAKE_INSTALL_PREFIX=xxx to install GROMACS to a non-standard location (default /usr/local/gromacs)
-DBUILD_SHARED_LIBS=off to turn off the building of shared libraries

-DGMX_FFT_LIBRARY=xxx to select whether to use fftw, mkl or fftpack libraries for FFT support
-DCMAKE_BUILD_TYPE=Debug to build GROMACS in debug mode

It's very important you use the options GMX_GPU and GMX_USE_OPENCL.

If the OpenCL files cannot be found, you could try to specify them (and let us know, so we can fix this), for example:
cmake .. -DGMX_BUILD_OWN_FFTW=ON -DGMX_GPU=ON -DGMX_USE_OPENCL=ON
-DCMAKE_BUILD_TYPE=Release
-DOPENCL_INCLUDE_DIR=/usr/include/CL/ -DOPENCL_LIBRARY=/usr/lib/libOpenCL.so
Then make and optionally check the installation (success currently not guaranteed). For make you can use the option "-j X" to launch X threads. Below is with 4 threads (4 core CPU):
make -j 4

If you only want to experiment, and not code, you can install it system-wide:
sudo make install
source /usr/local/gromacs/bin/GMXRC
In case you want to uninstall, that's easy. Run this from the build-directory:
sudo make uninstall

Building on Windows, special settings and problem solving

See this article on the Gromacs website. In all cases, it is very important you turn on GMX_GPU and GMX_USE_OPENCL. Also the wiki of the Gromacs OpenCL project has lots of extra information. Be sure to check them, if you want to do more than just the below benchmarks.

Run & Benchmark

Let's torture GPUs! You need to do a few preparations first.

Preparations

Gromacs needs to know where to find the OpenCL kernels, for both Linux and Windows. Under Linux type: **export GMX_OCL_FILE_PATH=/path-to-gromacs/src/**. For Windows define GMX_OCL_FILE_PATH environment variable and set its value to be /path_to_gromacs/src/

Important: if you plan to make changes to the kernels, you need to disable the caching in order to be sure you will be using the modified kernels: set GMX_OCL_NOGENCACHE and for NVIDIA also CUDA_CACHE_DISABLE:
export GMX_OCL_NOGENCACHE
export CUDA_CACHE_DISABLE

Simple benchmark, CPU-limited (d.poly-ch2)

Then download archive "gmxbench-3.0.tar.gz" from ftp://ftp.gromacs.org/pub/benchmarks. Unpack it in the build/bin folder. If you have installed it machine wide, you can pick any directory you want. You are now ready to run from /path-to-gromacs/build/bin/ :
cd d.poly-ch2
../gmx grompp
../gmx mdrun
Now you just ran Gromacs and got results like:
Writing final coordinates.

	Core t (s)	Wall t (s)	(%)
Time:	602.616	326.506	184.6
	(ns/day)	(hour/ns)	

Performance: 1.323 18.136

Get impressed by the GPU (adh_cubic_vsites)

This experiment is called "NADP-DEPENDENT ALCOHOL DEHYDROGENASE in water". Download "ADH_bench_systems.tar.gz" from ftp://ftp.gromacs.org/pub/benchmarks. Unpack it in build/bin.
cd adh_cubic_vsites
../gmx grompp -f pme_verlet_vsites.mdp
../gmx mdrun
If you want to run from the first GPU only, add "-gpu_id 0" as a parameter of mdrun. This is handy if you want to benchmark a specific GPU.

What's next to do?

If you have your own experiments, ofcourse test them on your AMD devices. Let us know how they perform on "adh_cubic_vsites"! Understand that Gromacs was optimised for NVidia hardware, and we needed to reverse a lot of specific optimisations for good performance on AMD.

We welcome you to solve or report an issue. We are now working on optimisations, which are the most interesting tasks of a porting job. All feedback and help is really appreciated. Do you have any question? Just ask them in the comments below, and we'll help you on your way.