Available online at www.prace-ri.eu

Partnership for Advanced Computing in Europe

GROMACS on Hybrid CPU-GPU and CPU-MIC Clusters: Preliminary Porting Experiences, Results and Next Steps

Sadaf Alam, Ugo Varetto[a]

[a]*Swiss National Supercomputing Centre, Lugano, Switzerland*

Abstract

This report introduces hybrid implementation of the Gromacs application, and provides instructions on building and executing on PRACE prototype platforms with Grahpical Processing Units (GPU) and Many Intergrated Cores (MIC) accelerator technologies. GROMACS currently employs message-passing MPI parallelism, multi-threading using OpenMP and contains kernels for non-bonded interactions that are accelerated using the CUDA programming language. As a result, the execution model is multi-faceted where end users can tune the application execution according to the underlying platforms. We present results that have been collected on the PRACE prototype systems as well as on other GPU and MIC accelerated platforms with similar configurations. We also report on the preliminary porting effort that involves a fully portable implementation of GROMACS using OpenCL programming language instead of CUDA, which is only available on NVIDIA GPU devices.

1. Introduction to GROMACS and PRACE Prototype Systems with Accelerator Devices

GROMACS is a versatile package used to perform molecular dynamics, i.e. simulating the Newtonian equations of motion for systems with hundreds to millions of particles [1]. It is primarily designed for biochemical molecules like proteins, lipids and nucleic acids that have a considerable number of complicated bonded interactions, but since GROMACS is extremely fast at calculating the non-bonded interactions (that usually dominate simulations) many research groups are also using it for research on non-biological systems, e.g. polymers.

The PRACE prototypes and the target systems that are considered for this study include clusters at CSC (Finland), CINECA (Italy), PSNC (Poland) and CSCS (Switzerland) [2,3,4,5]. These systems contain either the GPU or MIC accelerator devices or both [6,7,8]. Details of the target platforms are provided in the subsequent sections.

- The goals of the project
  - To establish whether GROMACS can be ported for the PRACE prototype systems with a variety of accelerator devices
  - To identify system parameters that influence efficiency of the application for the prototype system
  - To project performance and scaling considerations for a hypothetical tier-0 system, which may be based on the prototype technology
- Work done in the project, including
  - Analysis of hardware and software dependencies for GPU accelerated version of GROMACS
  - Evaluation of performance and correctness with different systems
  - Investigation into the limitations of portable versions of GROMACS using OpenCL [9] implementation of OpenMM package [10]

- Benchmarking GROMACS on GPU accelerator systems with different node configurations

- Results obtained
  - GPU accelerators improve efficiency of GROMACS for systems with single and multiple accelerator GPU devices
  - Scaling efficiencies have also been observed over multiple nodes and results are compared for IB QDR and FDR interconnects
  - OpenCL version of GROMACS needs further investment into the code development efforts
  - GROMACS would rely on implementation of non-bonded calculations in MIC intrinsics
  - Only offload implementation of GROMACS on the current generation of MIC devices is likely to yield scaling efficiencies on Tier-0 scale platforms

- Conclusions and summary
  - A single node system with an Nvidia Kepler GPU and an Intel SandyBridge CPU is about 1.3 times faster than a node with two SandyBridge GPUs for a 512k water molecule dataset.
  - On multi-node systems scalability (up to 16 nodes) is similar for both CPU and GPU versions. Performance efficiency is 2x on the GPU accelerated version
  - Further investigation is needed to improve the scaling efficiencies onto large, tier-0 scale systems.
  - Performance slowdown has been observed on Intel Xeon Phi or MIC for both native and symmetric modes of execution in absence of a platform specific tuned implementation.

2. Hybrid GROMACS Implementation

In this white paper, we consider three different programming and execution models for hybrid GROMACS porting and evaluation on PRACE prototype systems:

- MPI and OpenMP for Intel Xeon Phi or MIC in native and symmetric execution modes
- MPI, OpenMP and CUDA on systems with Nvidia Kepler accelerators
- With OpenMM package that has OpenCL implementation for system with AMD GPU devices

The version of GROMACS tested on PRACE prototype systems is 4.6.3. Experimental versions of GROMACS eventually merged with 4.6 were used on various CSCS prototypes.

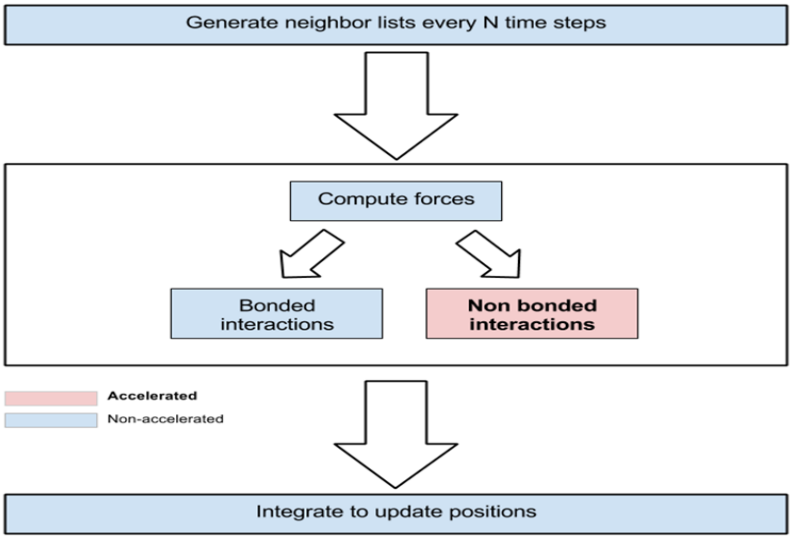

Figure 1: GROMACS compute-flow diagram. Adopted from [11].

Figure 1 shows the compute flow of a typical GROMACS run, the only part ported to GPU acceleration is the non-bonded interaction computation. The following CUDA header and source files have been generated for the GPU implementation in the Gromacs source tree (`nbnxn` is for non-bonded algorithms), including the source lines of code (SLOC):

- `src/mdlib/nbnxn_cuda/nbnxn_cuda_kernel_legacy.cuh` (255)
- `src/mdlib/nbnxn_cuda/nbnxn_cuda_kernel.cuh` (311)
- `src/mdlib/nbnxn_cuda/nbnxn_cuda_kernels.cuh` (38)
- `src/mdlib/nbnxn_cuda/nbnxn_cuda_kernel_utils.cuh` (225)
- **`src/mdlib/nbnxn_cuda/nbnxn_cuda.cu` (453)**
- **`src/mdlib/nbnxn_cuda/nbnxn_cuda_data_mgmt.cu` (682)**
- `src/gmxlib/cuda_tools/vectype_ops.cuh` (113)
- `src/gmxlib/cuda_tools/cudautils.cuh` (71)
- **`src/gmxlib/gpu_utils/memtestG80_core.cu` (577)**
- **`src/gmxlib/gpu_utils/gpu_utils.cu` (510)**
- **`src/gmxlib/cuda_tools/cudautils.cu` (151)**
- **`src/gmxlib/cuda_tools/pmalloc_cuda.cu` (45)**
- **`src/gmxlib/cuda_tools/copyrite_gpu.cu` (19)**
- **`src/contrib/openmm_gpu_utils.cu` (68)**

Unlike, the CUDA implementation, the Intel MIC implementation can be built natively *without modifying a single line of code*. However, this approach has severe performance implications. Non-bonded interactions in Gromacs have been tuned for different CPU targets using platform specific intrinsics. At compile time, users select an appropriate GMX_CPU_ACCELERATION value, which could be SSE2, SSE4.1, AVX_128_FMA, AVX_256, None and others. Without using platform specific implementations (by selecting an optimal GMX_CPU_ACCELERATION in cmake) and by using the reference version (GMX_CPU_ACCELERATION=None) the code could be 3 to 10 times slower. Since the Intel Xeon Phi or MIC has a different ISA (Instruction Set Architecture) as compared to Intel Xeon x86_64 ISA; the SSE and AVX accelerated implementation cannot be built natively for the Intel Xeon devices. In order to avoid slowdown using the reference implementation, users need to develop implementation using Intel Xeon Phi intrinsics.

Build configuration

GROMACS is built with the CMake makefile generator tool; we used CMake 2.8.x versions to build GROMACS on all the platforms. The specific changes to the CMake environment required to build GROMACS with GPU, MIC and MPI support are listed below.

NVIDIA Kepler

- Use FFTW version 3 as the fft library:
  - specify the fftw library and include path in the FFTWF_* parameters
  - set the GMX_FFT_LIBRARY to fftw3
- Enable/disable MPI by setting the GMX_MPI flag to ON or OFF
- Add the –arch=sm_35 switch to the CUDA_NVCC_FLAGS parameter
- Enable GPU: set GMX_GPU to ON
- Set the GMX_CPU_ACCELERATION parameter to either SSE4.1 or AVX_256

Intel MIC

- Build FFTW for mic (alternatively try using MKL FFTW, performance difference is negligible)
- Build the code editing CMAKE_CXX_FLAGS to include –mmic in cmake settings
- In cmake, choose the reference implementation (GMX_CPU_ACCELERATION:STRING=None)
- The Sandy Bridge build with GMX_CPU_ACCELERATION:STRING=SSE4.1 is used without any changes for the symmetric execution

OpenMM is a library for providing tools for modern molecular modeling simulation. Initial support for GPU acceleration in GROMACS was available through the GPU port of the OpenMM library, which has been implemented in CUDA and OpenCL. However, the support of OpenMM has been deprecated in latest release version, 4.6.x.

3. Target Systems: PRACE Prototypes and Clusters with Accelerator Devices

Porting and evaluation of GROMACS have been targeted on the PRACE prototype systems with accelerator devices. These include:

- Advanced Scalable Hybrid Architecture (Scalable Hybrid) from CSC
- European Many Integrated Core Architecture (EURORA) from CINECA
- Assessment of Hybrid CPU/GPU Architecture (Fusion) from PSNC

Only key characteristics features that are necessary to understand why a different porting and mapping approach is necessary for these three systems have been highlighted in the paper. Figure 2 shows three types of nodes designs that are representative of the node architecture of these platforms. A CPU could be any x86_64 multicore system such as an Intel Sandy Bridge or AMD Opteron. An ACC could be a GPU or MIC accelerator device from any vendor: Intel, Nvidia or AMD. Two types of memory technologies could be present. DDR memories could be DDR3 for CPU or GDDR5 for accelerator devices or it could be a unified memory. Although the three prototypes exhibit distinct network characteristics, the scope of the white paper is limited to the node acceleration. Hence details on the network specifications and topologies are not provided in this report.

Node design A shows a node with dual-socket CPU and dual accelerator devices. The two CPU sockets are connected with a memory interface, for example, QPI for Intel Sandy Bridge processors, resulting in non-uniform memory architecture (NUMA). There are two accelerator devices, which are connected to each CPU, hence their distances from the network devices are different. The EURORA prototype system exhibits node design A. CSC phase I system is similar but has a single accelerator device. The system should contain two Intel MIC (Knights Corner) devices, each with more than 50 cores. The system has DDR3 memory for the CPU and a higher bandwidth GDDR5 memory for the MIC devices.

Node design B is similar to node design A except only one CPU socket and accelerator device is present on each processing node. The Scalable Hybrid prototype is expected to contain 50% of nodes with Intel KNC cards and 50% with Nvidia Kepler K20 devices.

Node design C represents a node of the PSNC Hybrid CPU/GPU prototype. This design has a few unique features. Firstly, there is a single memory that can be accessed by both the CPU and the accelerator device. The CPU is an AMD Opteron and GPU is an AMD Radeon device. The system contains a unified northbridge unit that manages and schedules memory access from CPU and GPU devices.

In addition to the PRACE prototype systems, a number of experiments are performed on an in-kind system at CSCS. This system is composed of node design A nodes, which can also be configured to exhibit characteristics of node design B. The system was composed of Intel Sandy Bridge CPUs and Intel KNC 5110p and Nvidia K20 accelerator devices.

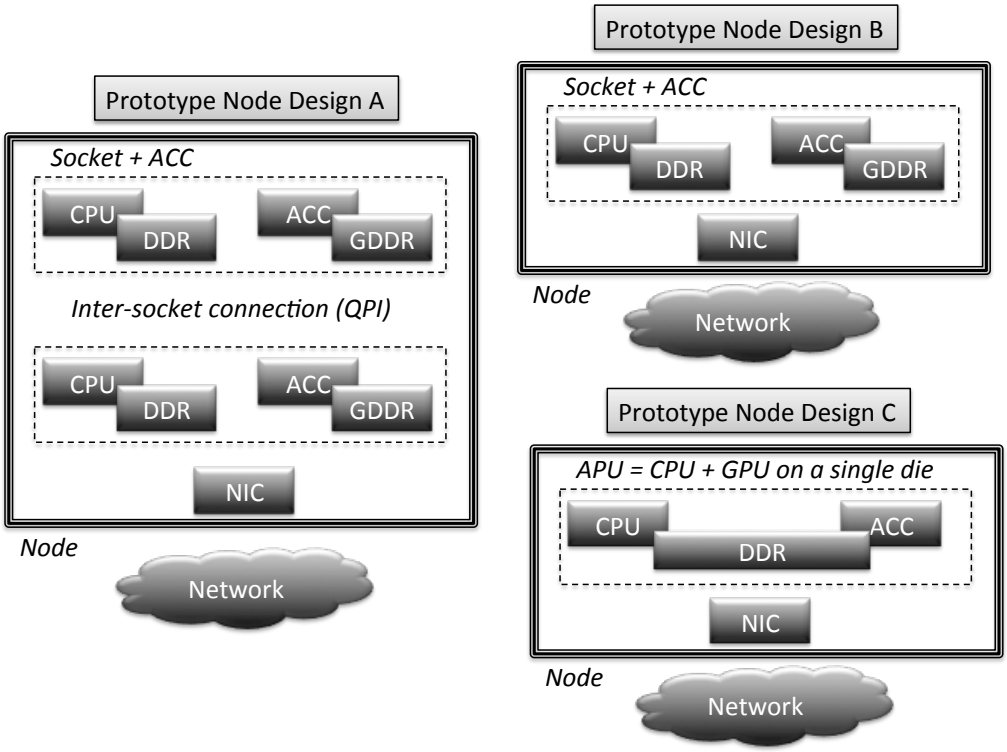

Figure 2: Representation of three types of nodes that are proposed for the PRACE accelerator based prototype systems

4. Results

Performance data is collected for the standard 1536k-water benchmark using the PME and RF solver methods [12]. The reported performance numbers are always in ns/day.

Kepler K20s on EURORA:

Single node runs with the RF method

Threads per node/CPU acceleration	*CPU*	*GPU*
16/SSE4.1	1.256	1.750
16/AVX_256	1.348	1.724
32/AVX_256	0.766	1.932

Multi-node (16 threads per MPI process 2 MPI processes per node, AVX_256) with the RF method

# MPI Processes	*CPU*	*GPU*
4	1.811	3.872
8	2.738	4.906
16	4.949	8.523

Intel MIC

Execute instructions for the symmetric mode on the CSC system:

```
> export MIC_PPN=4
> export MIC_OMP_NUM_THREADS=4
> export OMP_NUM_THREADS=8
> srun -n 1 mpirun-mic -m /path-to-cpu-exe/mdrun_mpi -c /path-to-mic-exe/mdrun_mpi
```

Intel MIC and Xeon experiments have been performed for the PME method.

Native mode with 32 threads: 0.106 ns/day

Symmetric mode with 8 threads on Sandy Bridge, one MPI task and with 4 MPI tasks 4 threads each on Intel MIC: 1.121 ns/day.

5. Summary and Suggestions for Future Work

The hybrid implementations of GROMACS MPI, OpenMP and CUDA versions together with platform specific intrinsics offer scalability and acceleration for Nvidia GPU based PRACE prototypes. OpenCL version of OpenMM could only be used for an older distribution and we were not able to execute the code with OpenCL 1.1. Note that only OpenCL 1.1 is available on the Nvidia GPU systems. OpenCL 1.2 is available as part of the Intel SDK and was made available recently on Intel Xeon Phi (MIC). We did not attempt to build the OpenCL version using the Intel SDK because of the dependency on an older version of the GROMACS distribution. Progress can be made if the GROMACS community reintroduces support for OpenMM in current and future versions of Gromacs distributions. Alternatively, CUDA implementation of non-bonded interactions in the current version of GROMACS could be rewritten in OpenCL to support a range of portable CPU and GPU devices.

GROMACS performance and scaling is highly sensitive to the implementation and selection of the non-bonded acceleration code (GMX ACCELERATION), number of OpenMP threads per node, the number of MPI tasks per node and mapping of MPI and OpenMP on the Intel MIC cards. CUDA version exhibits higher performance and scaling efficiencies on a small number of nodes. However, as the node efficiencies increase for the hybrid CPU and GPU implementation, we observe a slowdown in scaling efficiencies. Additional results can be found in [13], which could be reproduced on PRACE prototypes with Nvidia GPU devices. Experiments with different values of GMX_CPU_ACCELERATION demonstrate that there is a significant difference between the reference implementation and platform-tuned versions. Note that the reference implementation was used for building native and symmetric version for the MIC cards as MIC and x86_64 intrinsics are not compatible.

In order to improve scaling and performance efficiencies of GROMACS on PRACE prototypes and Tier-0 systems with accelerator devices, the following modifications to the code are required:

- Implementation of non-bonded interactions and FFT in MIC intrinsics
- Offload implementation for non-bonded interactions. FFT can be performed on the host in the offloaded mode if data transfer overheads a large over the PCIe interface
- OpenCL implementation within GROMACS (could be modified version of existing CUDA implementation)
- Investigate MPI and OpenMP affinity for MIC with a tuned version of kernels

In addition to the code modifications, we note that the inter-node communication bandwidth could influence scalability of the code. For example, the experiments on a QDR based system with blocking network topology may not scale similar to an FDR network with non-blocking connections.

Acknowledgements

This work was financially supported by the PRACE project funded in part by the EU's 7th Framework Programme (FP7/2007-2013) under grant agreement no. RI-283493. The work was achieved using the PRACE Research Infrastructure resources at CSC, PSNC, CINECA and CSCS.

References

[1] GROMACS: http://www.GROMACS.org/
[2] EURORA: http://www.hpc.cineca.it/hardware/eurora
[3] CSC prototype: https://confluence.csc.fi/display/HPCproto/HPC+Prototypes
[4] PSNC: http://www.man.poznan.pl/online/en
[5] CSCS Nvidia K20 and Intel Xeon Phi clusters: http://user.cscs.ch/hardware/dom_gpu_cluster & http://user.cscs.ch/hardware/dommic_intel_mic_cluster
[6] Intel Xeon Phi (MIC): http://www.intel.com/content/www/us/en/processors/xeon/xeon-phi-detail.html
[7] APU: http://www.amd.com/us/products/technologies/apu/Pages/apu.aspx
[8] CUDA: http://www.nvidia.com/object/cuda_home_new.html
[9] OpenCL: http://www.khronos.org/opencl
[10] OpenMM: https://simtk.org/home/openmm
[11] Porting GROMACS Molecular Dynamics Code to the Cell Processor: http://www.cs.unc.edu/~olivier/pdsec07.pdf
[12] Benchmark data: ftp://ftp.cscs.ch/out/uvaretto/water-1536k.tar.gz
[13] http://www.nvidia.com/docs/IO/122634/GROMACS-benchmark- report.pdf