ICHEC's GPU research: porting of scientific application on NVIDIA GPU

Mr Girotto Ivan, ICHEC consultant for Novel Architecture
Mr Yang Yang, ICHEC scholarship student

GPGPU (General Purpose computing on Graphics Processing Units) has become increasingly popular in the HPC community in recent years, where the GPU is now a viable component of new-generation compute platforms in addition to its traditional role in visualisation work. While it is not the only path towards widespread availability of peta-scale computing, it is currently the most promising one. ICHEC is pursuing the goals to port codes that are of major interests to the Irish scientific community to take advantage of GPGPUs. ICHEC will shortly carry on this work into the PRACE project where intend to play a key rule into the work-package dedicated to exploitation of accelerators for real applications.

Quantum ESPRESSO [2] is an integrated suite of computer codes for electronic-structure calculations and materials modeling at the nanoscale. It is based on density-functional theory, plane waves, and pseudopotentials (both norm-conserving, ultrasoft, and PAW).

The project started in 2002 as a DEMOCRITOS initiative, in collaboration with CIENCA and with research groups in Princeton University, Massachusetts Insitute of Technology (MIT) and Ecole Polytechnique Fdrale de Lausanne (EPFL).

Quantum ESPRESSO stands for opEn Source Package for Research in Electronic Structure, Simulation, and Optimization. It is freely available to researchers around the world under the terms of the GNU General Public License. The relevance of this code has been recently underlined, as it is part of the list of application benchmark suite delivered through the most important European project in HPC (PRACE and DEISA). These applications have been chosen after evaluation through industrial and scientific usage and scalability potential for running on petascale systems and beyond.

The software package interests big community of user and it's currently used by number of world-class research group spread world-wide such as Quasiamore Group @ MIT, Ermes @ NCUS, Tyndall National Institute in Ireland.

Code Porting

The earliest started work represents how it's possible to port real case of scientific application using CUDA software with a reasonable effort. Mainly bottleneck of the code PWscf are represented by 3DFFT, linear algebra (matrix multiplication), space integral and point function evaluation. The code is mostly implemented in Fortran which it is not "natural" portable on CUDA as well as C language. Such as we needed to implement wrappers for the CUDA code but the number of information available on web still help to speed-up this unfriendly process giving up frequently mistakes.

The current CUDA software available allowed us to cover the first two points. As described in [1] we implemented a wrapper which permit to catch all the [DZ]gemm calls performing them between GPU and CPU, this library it's fully described in the beside section. The same schema to overlap computation between CPU and GPU has been implemented for the 3D FFT. Unfortunately, we can reach poor improvement on the FFT while on the [DZ]GEMM we can get a significant gain on the whole application. The FFT routine, in the code, is called to trasfrorm wave-functions, charge density and potentials back and forward between reciprocal and real space. To optimize this operation, Quantum ESPRESSO adopt an ad-hoc FFT algorithm. The algorithm takes advantage of the fact that a 3D FFT is a linear superposition of three subsequent series of 1D FFTs along the Cartesian coordinates and for each series, only those 1D FFTs contains non zero elements are perfomed. The currently CUDA software available doesn't permit to directly port this implementation as it doesn't support FFT with stride, which should be implement in the next release. For this reason the FFT on the ported code perform FFT3D on the whole grid. Moreover we ported on CUDA two computational kernel that are remarkable for the test case dataset. Both the subroutines are related to the ultrasoft pseudopotential, as such pseudopotential which permits to distinguish the electronic charge density into an hard component (invariable) and a soft component which depends by the environmental interaction. The *addusdens* routines add the charge density related to the hard component of the pseudopotential at electron charge density in order to obtain the total charge density.

The kernel *newd* compute a factor of the potential of iteration between ion and electrons related to the component in semi-local form of the pseudopotential.

The work shortly described above permits to obtain the performance shown on Figure 1. The same results are obtained with both number of K points or with K = 0.

All this work is still performed respect the serial version of the PWscf code. We aim to work soon on the parallel version in order to assess the performance achievable also on hybrid architecture using CUDA software over the MPI parallelization. In this stage will come more and more important the porting of the FFT.

The modular structure of Quantum ESPRESSO package should make possible to adopt all the solutions adopted in this porting to other codes of the suite, at least at the code CP which deals with basic Car-Parrinello simulations.

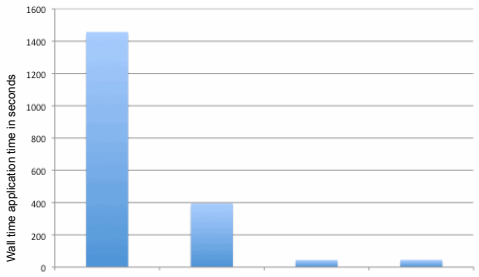

Figure 1 – Benchmark of a medium size input for PWscf, taken DEISA benchmark. AUSURF112 is a dataset for a gold surface of 112 atoms. The chart show results obtained on Fermi architecture compared with a compute node with two 2.8GHz Intel (Nehalem EP) Xeon X5560 quad-core processors.

PHIGEMM, THE GENERAL MATRIX MATRIX MULTIPLICATION LIBRARY ON CPU-GPU HYBRID ARCHITECTURE

Philip Yang

Irish Centre for High-End Computing

1. INTRODUCTION

The ϕGEMM library implements three of the BLAS3 General Matrix Matrix Multiplication functions: SGEMM (single precision), DGEMM (double precision) and ZGEMM (complex double precision). It took advantage of the underlying blas kernel functions on both CPU and nVidia CUDA based GPU.

2. ALGORITHM

The function computes the following $C = \alpha AB + \beta C$. Computation is performed on CPU and GPU concurrently in order to achieve high throughput. Due to the memory limit on device, if the portion assigned to GPU cannot fit in GPU memory, we will recursively split the matrix and perform a series of function calls with smaller input size.

2.1. Padding

Most underlying gpu blas kernels achieve high performance when the matrices are divisible by certain power of 2. The library apply padding of zeros when the dimension of input matrix is not divisible by some particular power of 2 which depends on the implementation of the underlying gpu blas kernels. This strategy makes the performance consistent over different input dimensions. Significant performance gain is manifested in some cases.

2.2. Page-Locked Memory

When problem size is small, the time is dominated by memory transfer. Thus hiding the memory latency is the most important part to guarantee the performance of small input size. For example, with input size of 1000, the memory transfer takes almost half of the execution time.

Page-locked memory could sustain a data rate around 5GB/s while ordinary memory could only sustain a data rate of around 3GB/s. The advantage of page-locked memory is especially obvious with small input size. However, to use page-locked memory, the user has to explicitly declare it. This is somehow hard for legacy code.

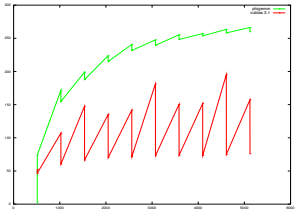

Fig. 1. Performance of phigemm library with thuncking wrapped cublas 3.1 library. Green line for the case with phigemm library, red line for the case with cublas 3.1 library. *The labels in this plot are too small.*

3. USAGE

The user are not obliged to have any background in CUDA programming. The functions are encapsulated in a shared library. To use them, the user should link her program to the library. Prior to all the GEMM calls, the user should call the initialization routine to start the library. Upon finish using the library, a shut-down routine should be called. Notice that if the shut-down routine is not invoked, the memory allocated by the library (if any) could not be cleaned.

The user could choose to manually tune the performance by manually adjust several runtime parameters such as determing which matrix to split and the portion assigned to gpu by manipulating environment variables.

REFERENCES

[1] M. Fatica, "Accelerating linpack with CUDA on heterogenous clusters" in Proc. of 2nd Workshop on General Purpose Processing on Graphics Processing Units, 2009.

[2] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, R. Car, C. Cavazzoni, D. Ceresoli, G. L. Chiarotti, M. Cococcioni, I. Dabo, A. Dal Corso, S. Fabris, G. Fratesi, S. de Gironcoli, R. Gebauer, U. Gerstmann, C. Gougoussis, A. Kokalj, M. Lazzeri, L. Martin-Samos, N. Marzari, F. Mauri, R. Mazzarello, S. Paolini, A. Pasquarello, L. Paulatto, C. Sbraccia, S. Scandolo, G. Sclauzero, A. P. Seitsonen, A. Smogunov, P. Umari, R. M. Wentzcovitch, J.Phys.:Condens.Matter, 21, 395502 (2009)

AKNOWLEDGEMENTS

This work has been realized thanks to important collaboration with Carlo Cavazzoni (CINECA Supercomputing Center), Layla Martin Sarmos (DEMOCRITOS), Rob Farber (NPL), Stan Tomov (MAGMA Group) and Massimiliano Fartica (NVIDIA)

Ireland's EU Structural Funds Programmes 2007 - 2013
Co-funded by the Irish Government and the European Union

HEA Higher Education Authority An tÚdarás um Ard-Oideachas

sfi science foundation ireland fondúireacht eolaíochta éireann

EUROPEAN REGIONAL DEVELOPMENT FUND