phiGEMM: a CPU-GPU library for porting Quantum ESPRESSO on hybrid systems

Filippo Spiga and Ivan Girotto
Irish Centre for High-End Computing (ICHEC), Dublin - Ireland
Outline

• Motivations
• Starting point: compute using libraries
• The CPU-GPU PHI GEMM library
• Serial GPU-accelerated PWscF
• Multiple GPUs 3D-FFT case-study
• What happened next

Feb 16, 12
Special Session on GPU Computing - PDP2012, Garching (DE)
Motivations & goals

As involved in PRACE 1-IP project, WP 7.5

- Application porting to GPU
  - through CUDA kernels
  - through CUDA libraries
  - through Directives (#)
- Evaluating new models/new paradigms

Main goals:
- Couple CUDA libraries and CPU libraries
- Create a library for transparently accelerate (part) of a complex application
- Hit, assess and discuss the bandwidth effects of CPU+GPU BLAS and FFT

→ PRIMARY GOAL: going to production!
The CPU-GPU PHIGEMM library/1

Functionalities

Starting points:
- GPU & CPU libraries → do not re-invent the wheel!
- NVIDIA GPU HPL (by M. Fatica)

The phiGEMM library...
- independent open-source project
- targets all GEMM operations (S,C,D,Z)
- C and FORTRAN interfaces
- CUDA 3.x(#) and CUDA 4.x compatible
- performs call-by-call profiling
- supports multiple-GPU (CUDA 4.x)
- current version 1.6(#)

4 Feb 16, 2012 Special Session on GPU Computing - PDP2012, Garching (DE)
The CPU-GPU PHI GEMM library/2
the idea behind the scene
The CPU-GPU PHI GEMM library/3
splitting on both sides

Full A, partial B and partial C on GPU

What about if they do not fit GPU memory?

Partial A, full B and partial C on GPU
The CPU-GPU PHI GEMM library/4 recursion for big matrices

Recursive splitting

Load-balancing penalty
The CPU-GPU PHI GEMM library/5
the splitting issue

It is all about load-balancing...
• try a guess → no good idea
• you know CPU and GPU peak perf → static approach
• through internal timing you can adjust the CPU/GPU work-load dynamically until a specific tolerance is achieved → dynamic approach

It is all about effective bandwidth...
• even if you want, you cannot guess it
• it is very variable from system to system
High-end workstation
• 2 six-cores Intel(R) Xeon(R) X5650 @ 2.67GHz
• 24 GBytes of HOST memory
• 2 Tesla C2050 with 3 GByte of DEVICE memory

During the tests...
• We used both one socket (6 cores) and two sockets (12 cores)
• GPUs are on independent PCI channels
• We “manually” reduced the GPU memory available
The CPU-GPU PHI{GEMM} library

DGEMM, 12 OMP, full GPU memory

phi{GEMM} single- and multi-GPU performance

CUBLAS Thunking
CUBLAS peak
MKL

phi{GEMM} (1 GPU)
phi{GEMM} (2 GPU)

Dimension \([M=N=K]\)

Results of July 2011
The CPU-GPU PHI_GEMM library/7
ZGEMM, 6 OMP, 33% GPU memory (~1 GByte)

phiGEMM single- and multi-GPU performance

Results of Jan 2012
The CPU-GPU PHI GEMM library

DGEMM, m=n=k=25k, 12 OMP

Note that...

• Is there any BLAS CPU library faster then MKL? Just plug-it
• Is there any BLAS GPU library faster than CUBLAS? Just plug-it

Scaling is NOT linear
(as expected)
Quantum ESPRESSO

• Quantum ESPRESSO is an integrated suite of computer codes for electronic-structure calculations and materials modeling at the nanoscale.
  – Density-Functional Theory, plane waves, pseudo-potential, ...

• Very complex set of codes with shared modules
  – ~500K lines of code (FORTRAN and C)

• Parallel code that run on several HPC architectures
  – Linux clusters, CRAY supercomputers, IBM Power & BlueGene, NEC, ...

• It supports MPI and OpenMP

• Two main packages: PWscf and CP
Serial GPU-accelerated PWscf/1

- It **extensively** uses libraries
  - BLAS $\rightarrow$ phiGEMM (CUDA vs single-core, $\sim 5x$)
  - LAPACK$^\text{(#) )}$
  - FFTW (mainly in vloc_psi)
- It already well supports OpenMP
- Wall-time is mainly distributed within few routines
  - addusdens (CUDA vs single-core, $\sim 20x$)
  - newd (CUDA vs single-core, $\sim 7x$)
  - vloc_psi (CUDA vs single-core, $\sim 9x$)
- AUSURF112 is a “fair enough” starting point
Serial GPU-accelerated PWscF/2
a deep look inside: where libraries are
Serial GPU-accelerated PWscf/3 benchmarks - AUSURF112, serial

BONUS

Results of July 2011
The vloc_psi/3D-FFT issue

- The 3D-FFT has been recognized as a potential bottleneck for large calculations in all scientific DFT codes.
- BLAS and FFT have two different compute footprints (see the paper).

Q: is it worth to allow vloc_psi to exploit multiple GPU?
Q: how much I have to compute to compensate the data transfer?

In the parallel implementation vloc_psi acts over a distributed data, so the 3D-FFT grid is distributed (expected compulsory CPU ←→ GPU data transfers)

→ completely different strategy (GTC2012!!)
Multiple GPUs batch 3D-FFT case-study
performance evaluation

3D-FFT on multiple GPU

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>1 GPU</th>
<th>2 GPU</th>
<th>4 GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096 x 64³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 x 128³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 x 256³</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

3D-FFT on multiple GPU vs single-core FFTW

<table>
<thead>
<tr>
<th>Number of GPUs</th>
<th>1 GPU</th>
<th>2 GPU</th>
<th>4 GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>4096 x 64³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512 x 128³</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>64 x 256³</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

- We develop the phiGEMM library, a reliable library to handle CPU-GPU GEMM computations by “using” other libraries
  - [http://qe-forge.org/projects/phigemm](http://qe-forge.org/projects/phigemm)
- We develop a serial GPU-accelerated version of PWscf
  - [http://qe-forge.org/projects/q-e](http://qe-forge.org/projects/q-e) (PRACE branch)
- We assess performance and bottlenecks of phiGEMM running a complex application using real benchmarks (PWscf)
- We hit new limits and challenges
What happened/happening next

- Refinement of the splitting strategy and dynamic split factor
- Faced the PHI GEMM worst case scenario
  - very long k, totally different splitting strategy
- Multi-GPU CUDA kernels for PWscf serial calculation only
  - Yes but only where it is really worth (BLAS ok, FFT depends)
- Full Parallel GPU-accelerated PWscf (MPI+OpenMP+CUDA)
  - transform distributed computations in local computations to use GPU

Feb 16, 2012
Special Session on GPU Computing - PDP2012, Garching (DE)
Thank you for your attention!

QUESTIONS?