Accelerating industrial applications:

The development of basic GPU kernels for the new block AMG algorithms for solving SLE with explicitly calculated sparse basis

Afanasyev Ilya[1], Potapov Yury[1], Sergey Kharchenko[2]
Lomonosov Moscow State University[1], Tesis[2]

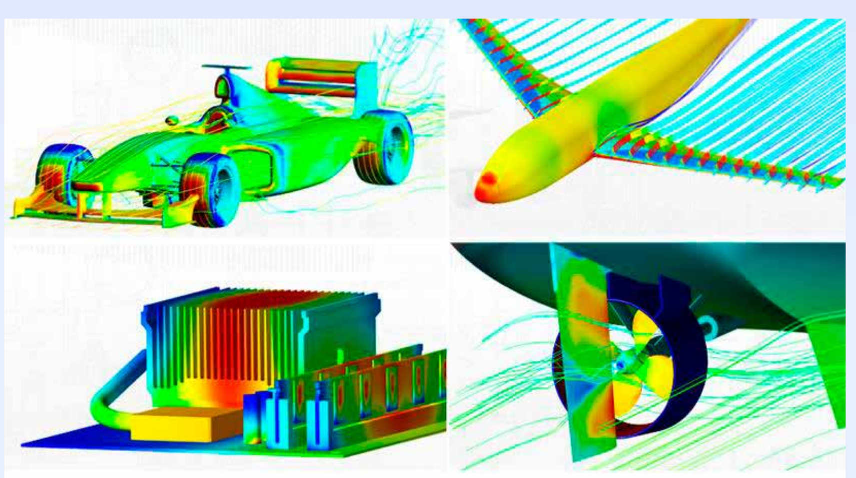
1. *Introduction*

AMG (Algebraic MultiGrid) algorithms can be applied to a wide range of applications:

- hydrodynamics
- mechanics
- electromagnetism

To solve problems of hydrodynamics there is software package FlowVision - an integrated multi-purpose solution for three-dimensional modeling of flows created by the development team from the Russian company Tesis. FlowVision is based on the numerical solution of three-dimensional steady and unsteady equations of fluid dynamics and gas.

However FlowVision supports the calculation only on high-performance multicore CPU clusters. The main idea of this work - to develop a computational kernels, which allow efficient GPU using



4. *Why not cuBLAS/cuSPARSE?*

These operations are supported by cuBlas and cuSparse libraries. However, they are not suitable for us because of the following reasons:

- they use different from FlowVision ways to store matrices
- have a poor support of running a large number of small tasks
- don't have optimizations for small dense matrix sub-blocks

5. *Hardware testbed*

- GPU: NVIDIA Tesla K20c, Tesla K40, GEFORCE TITAN
- CPU: Intel Xeon E5620, Intel Dual Core
- OS: Ubuntu version 12.04
- CUDA: Version 6.5
- Compiler: nvcc 6.5 & gcc 4.8 with –O2

2. *The goal of our work*

To transfer part of computational functions of package FlowVision to GPU it is necessary to implement CUDA kernels of the following basic linear algebra operations:

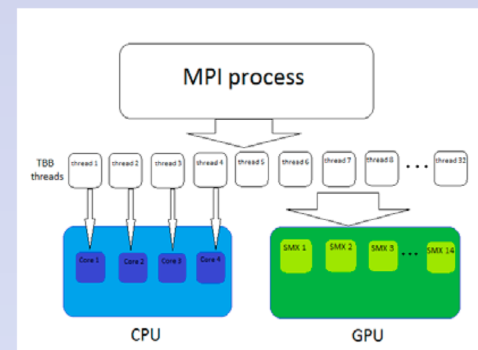
- DAXPY
- dot product
- multiplication of a sparse block matrix and transpose to it on a block of dense vectors
- complete(or not) LU factorization of a sparse block matrix
- solution of block system with finely upper and lower triangular sparse matrix

The most important is to implement simultaneous and uniform usage of CPU and GPU resources.

It is also important to mention that the main direction of optimizations is small tasks, because current kernels are aimed at solving subproblems with maximum size of vectors/matrices approximately 50 000 – 100 000 float elements(however whole problem can be very huge).

3. *The main ideas of our work*

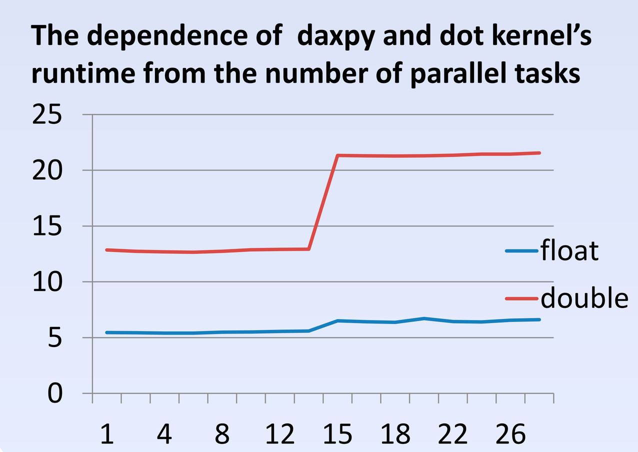
In order to simultaneously use the computing resources of the CPU and GPU the following programming mode is used:

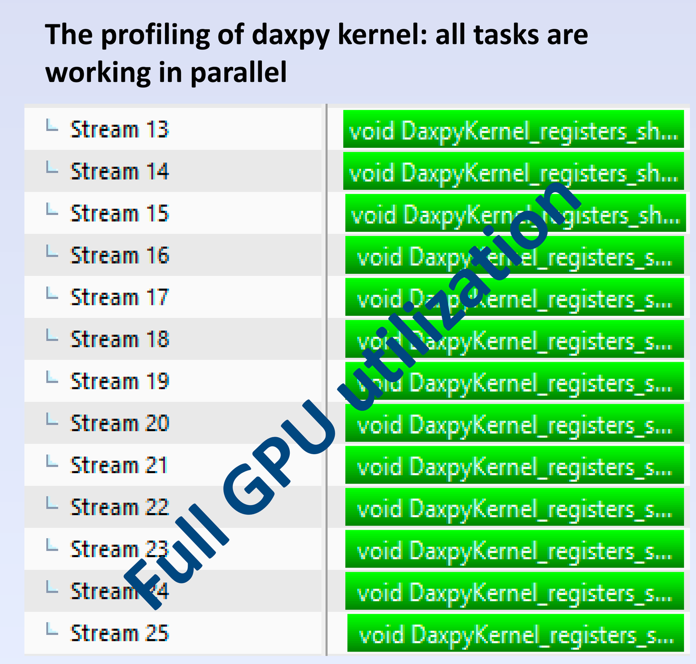


Each MPI process (corresponding to supercomputer's node) launches the group of threads, where synchronization is performed by Intel TBB.
Computational work within each thread may perform any CPU core or multiprocessor GPU.

In this context, in order to maximize performance and parallelism we must use GPU's architecture Kepler, allowing to run up to 32 concurrent kernels.

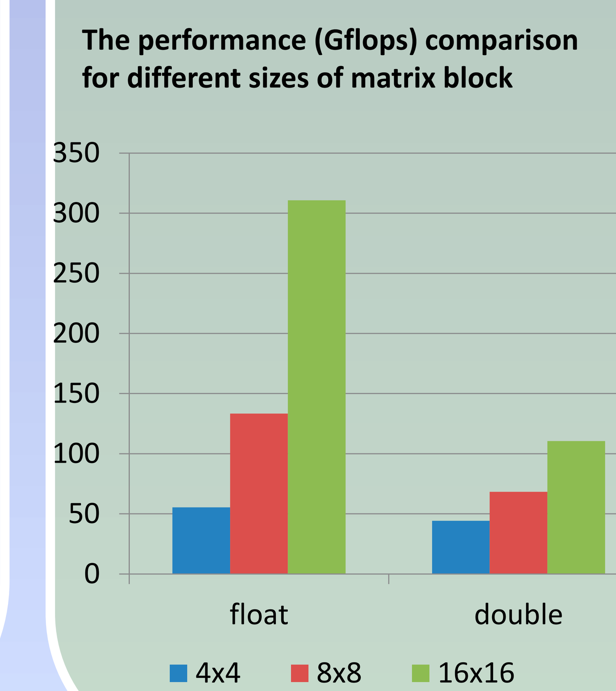
Thus on the GPU are launched about 14-32 parallel kernels, and each of one uses no more than one multiprocessor.

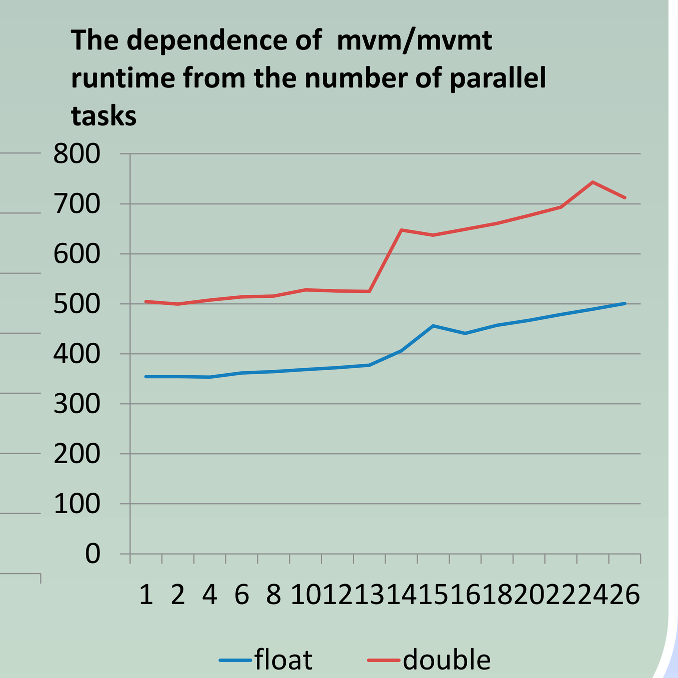




Sparse block matrices

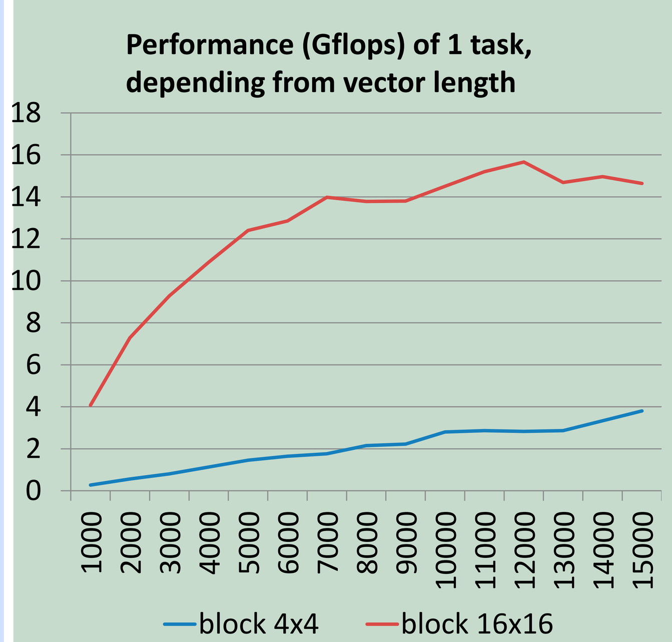
Multiplication of sparse block matrix in CRS format and transposed matrix in CSC format to a block of dense vectors (MVM and MVMT). The block size is in {1, 2, 4, 8, 16}
The main approach is to pre-commit permutation of matrix, so that the neighbour rows have the same number of blocks, and such groups of rows can be handled by independent small kernels. For transposed matrix we permute columns to gain groups with same amount of blocks.

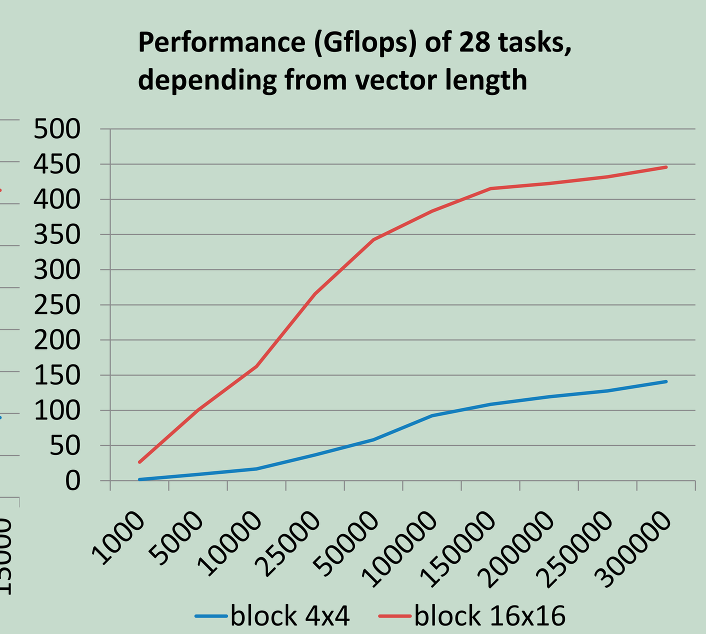


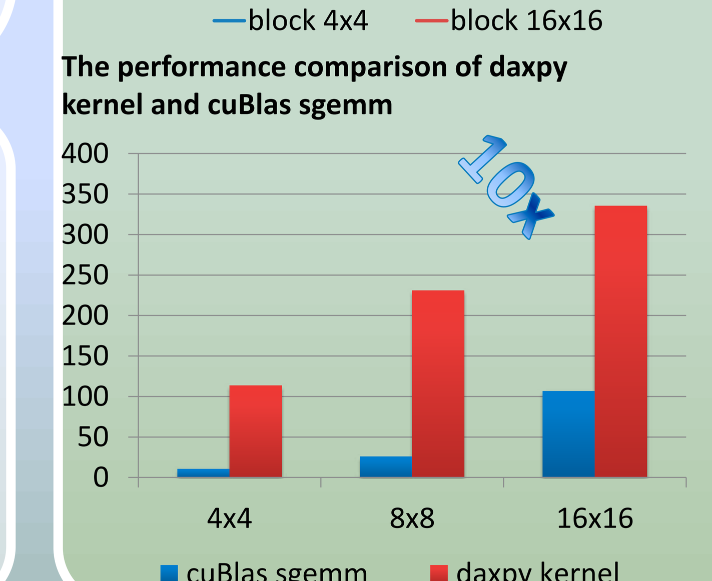


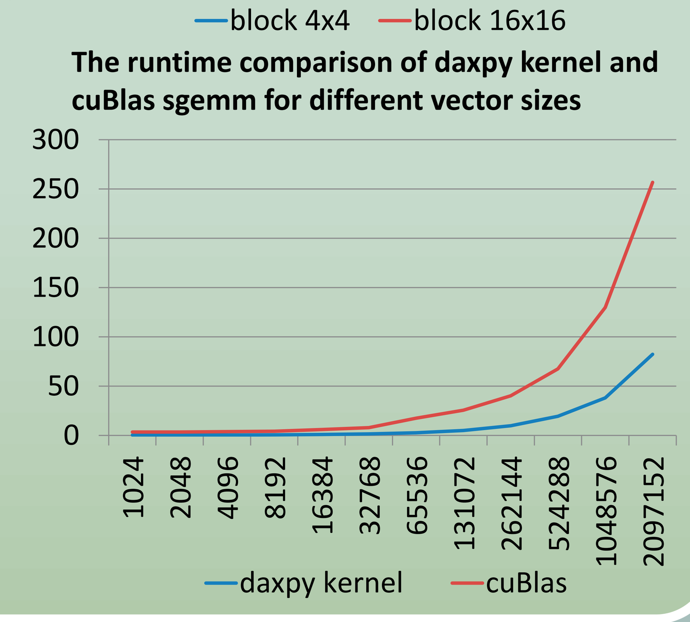
DAXPY

Block operation DAXPY is a straightforward block generalization of dot operation AXPY (Level 1 BLAS from Lapack):
Y + = X * A, where X, Y are rectangular matrixes M × N2 and M × N1 size, and A is small block size N1 × N2, where M >> {N1, N2}, N1 and N2 have one of the fixed sizes {1, 2, 4, 8, 16}



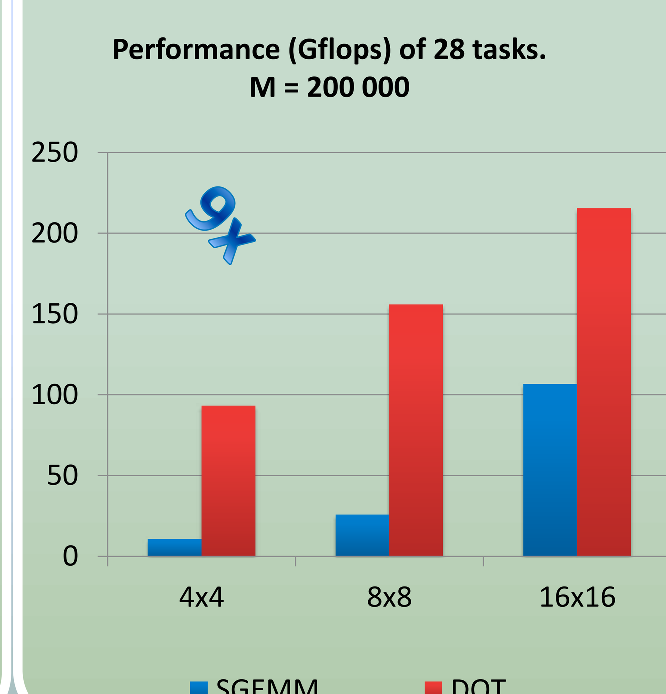


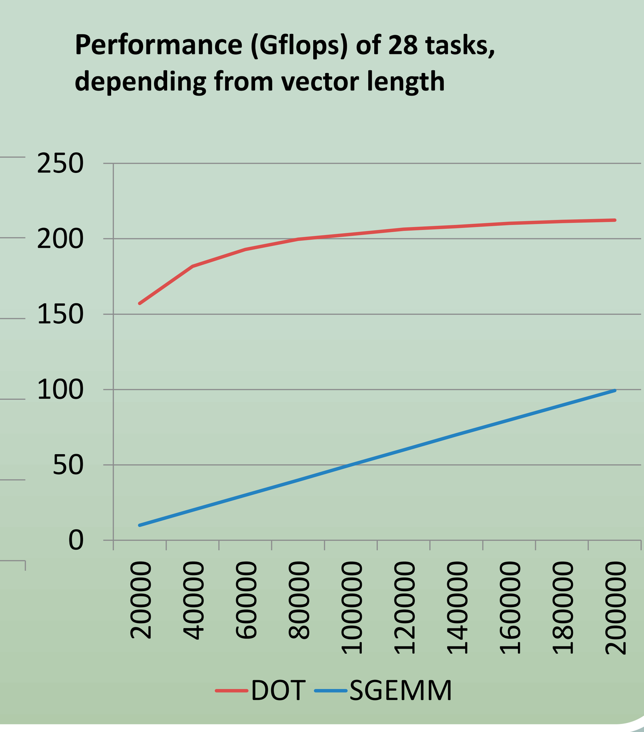




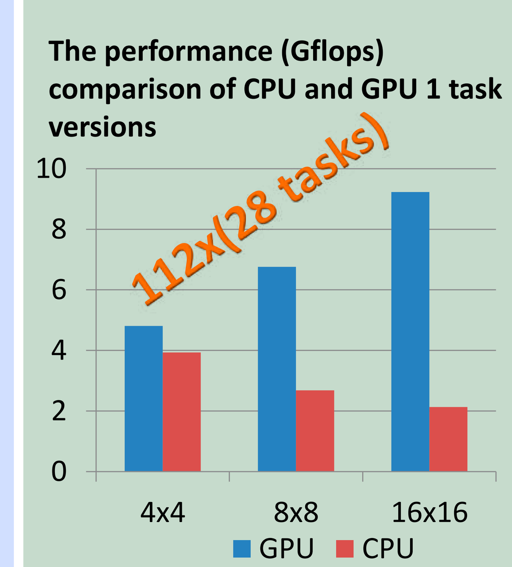
Dot product

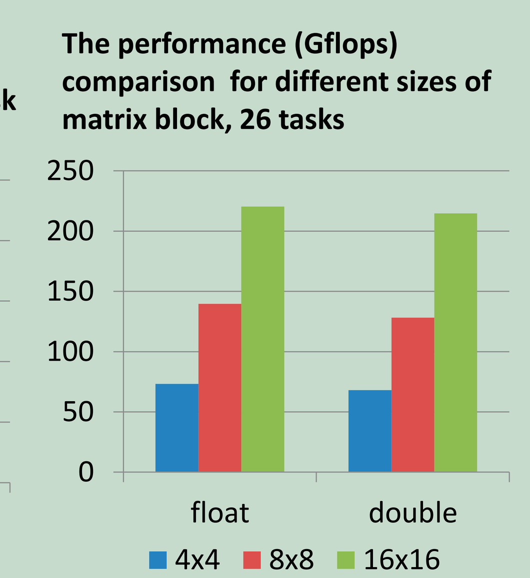
In both operations (dot and DAXPY) dense blocks are stored by rows to enable data locality.
Similar a BLAS DOT, Block DOT operation calculates scalar product. But instead of two vectors as an input , it receives two blocks of vectors and return as output one block of scalar products.
$C = A^T * B$.
A and B are matrices with sizes M x N_A and M x N_B.
M >> {N_A, N_B},
N_A and N_B have one of the fixed sizes {1, 2, 4, 8, 16}
Same result can be obtained by using a GEMM function from cuBLAS.



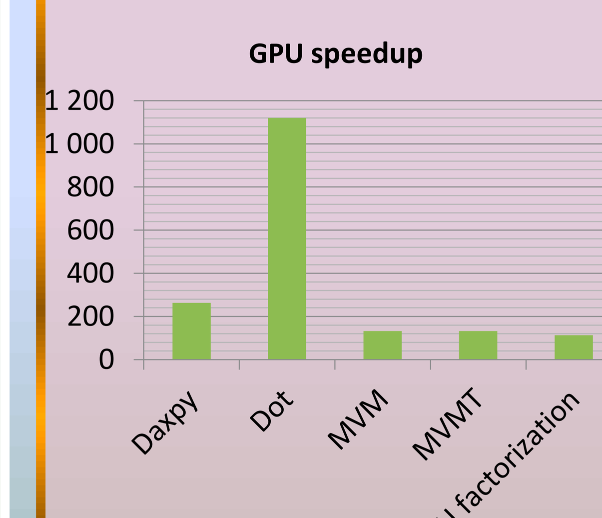


LU factorization





Conclusions



As a result of this work considered FlowVision computational functions have been accelerated from 50 to 1100 times relatively serial one-core versions

This material is based upon work supported by the Ministry of Education and Science of the Russian Federation (Agreement N14.607.21.0006, unique identifier RFMEFI60714X0006).