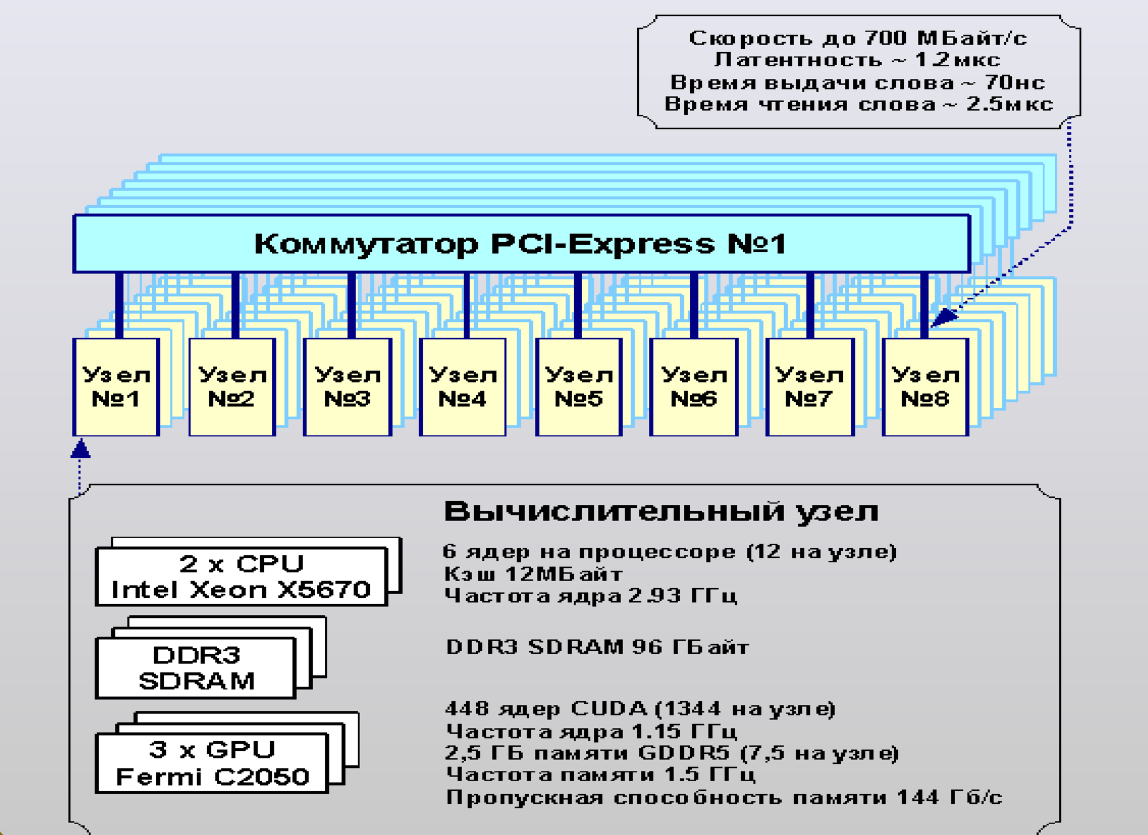
CATEGORY: DEVELOPER - ALGORITHMS - DA04
POSTER P5132
CONTACT NAME
Ivan Goryachev: kvant87@gmail.com

GPU TECHNOLOGY CONFERENCE

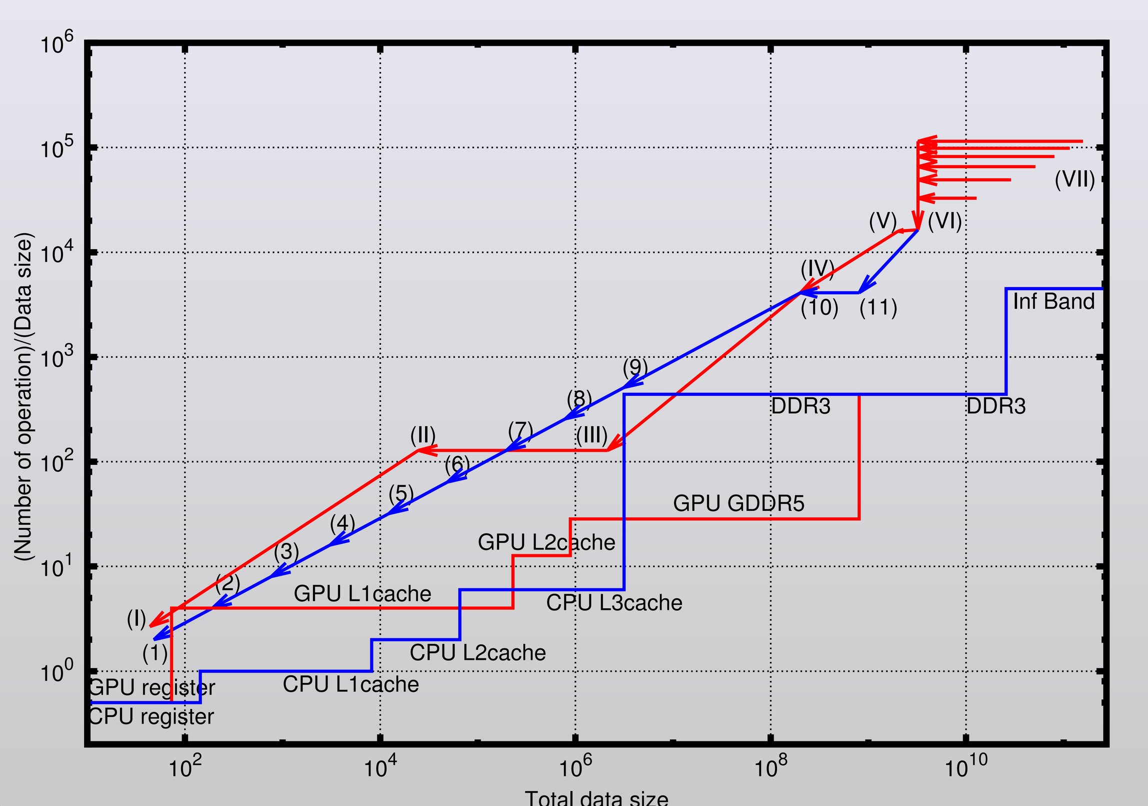
LRnLA Model of Computation for Geterogenious Calculations

Goryachev Ivan, Levchenko Vadim, Zakirov Andrey
Keldysh Institute of Applied Mathematics, Moscow, Russia

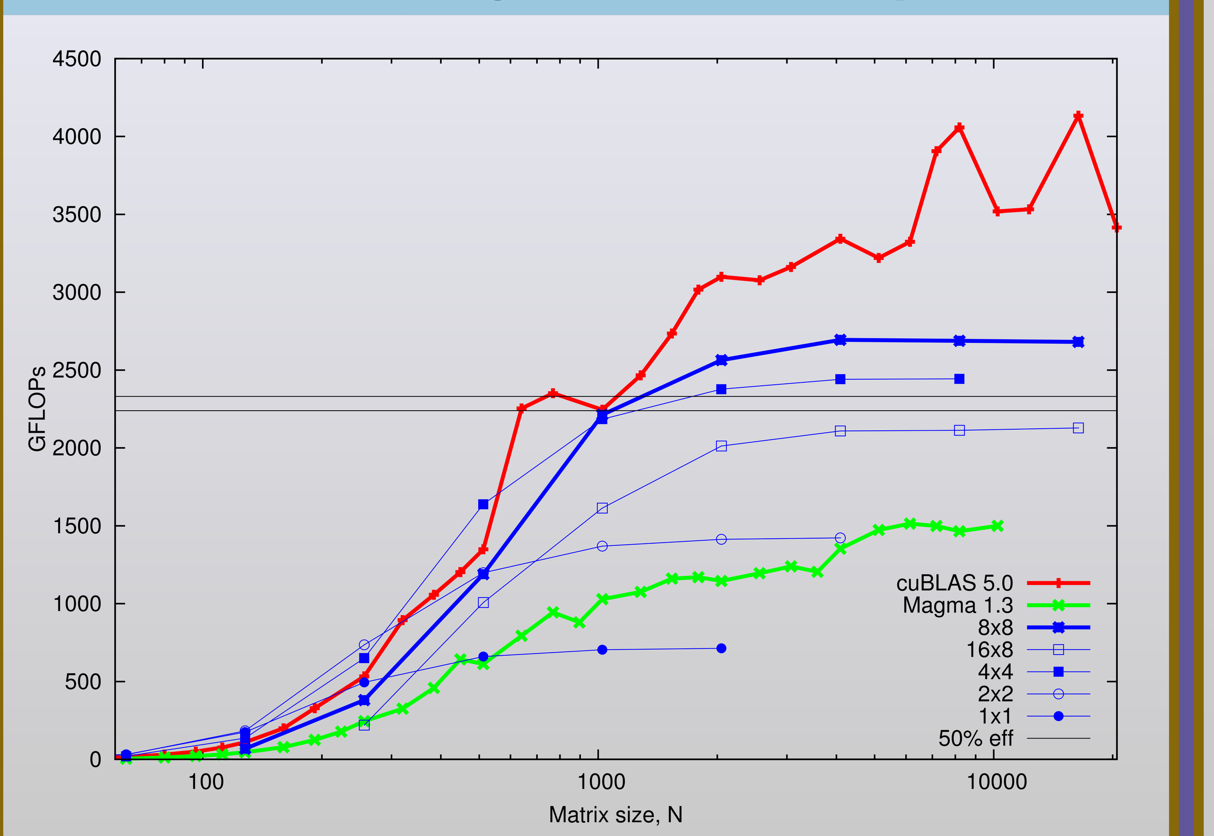
Cluster System K-100 [3]



LRnLA Model of Computation for general matrix multiplica



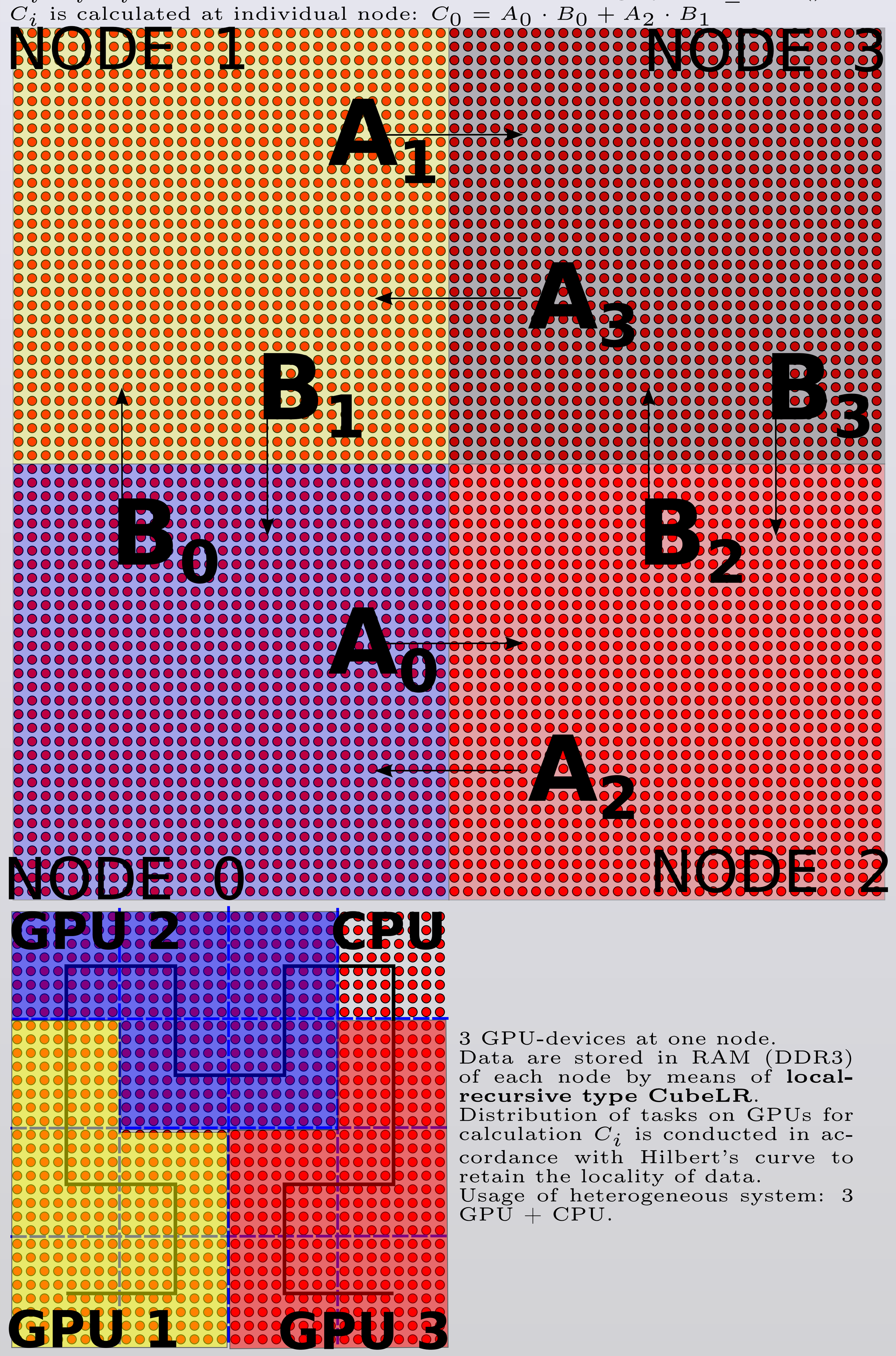
Result Performance for general matrix multiplication



Algorithmic principles for matrix multiplication

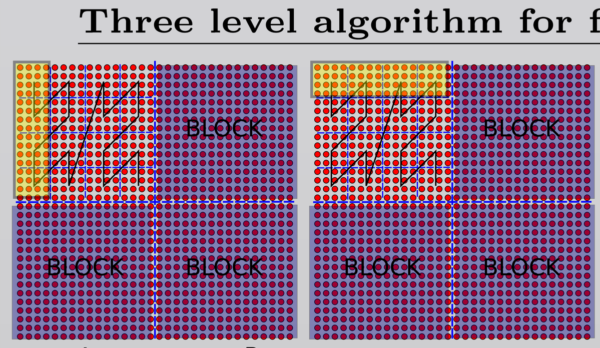
Five level algorithm of full matrix multiplication

Size of matrix: $N = (1 \div 7) \times 2^{15}$ (28GB per node)
Number of nodes: $1 : 4 : 9 : 25 : 36 : 49$
A_i, B_i, C_i initialize at individual node. Data sending by MPI_Isend();
C_i is calculated at individual node: $C_0 = A_0 \cdot B_0 + A_2 \cdot B_1$

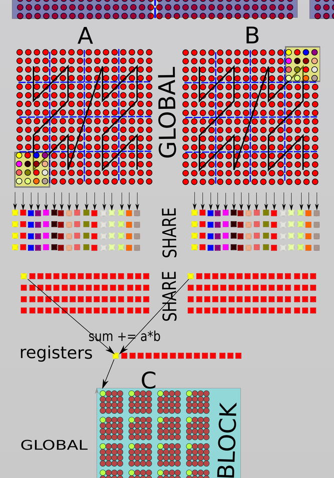


3 GPU-devices at one node.
Data are stored in RAM (DDR3) of each node by means of **local-recursive type CubeLR**.
Distribution of tasks on GPUs for calculation C_i is conducted in accordance with Hilbert's curve to retain the locality of data.
Usage of heterogeneous system: 3 GPU + CPU.

Three level algorithm for full matrix multiplication



Global memory volume of one GPU is 2560 MB
Data localized in global memory GPU are merged on substructures to localized them in cache memory.
Volume of L2 is 768 РЭРÇ.



Data are localized in memory of each SM and handled by different program-blocks (not CUDA-blocks!).
Program-block consists of CUDA-threads groups with size of
1x1 / 2x2 / 4x4 / 16x8 / 8x8,
that makes possible to utilize register memory.
CUDA-thread group splits into hardware structures named "half warp". It guarantees conflict-free access to the aligned address space of local memory.
Process of calculation is handled by local-recursive traversal CubeLR.

Similarities and differences between two problems

GEMM

Data volume: $D = 3N^2$
Number of operations: $O = N^3$
$O(D) = (1/3^{3/2})D^{1+\frac{1}{2}}$

$(O/D)(D)$:
Coefficient of locality$= N/3 = \sqrt{D/3}$.

FDTD

Data volume: $D = 6N^3$
Number of operations: $O = 30N^3 N_t$,
for $N_t > N$ let: $N_t = N \cdot N_T$
$O(D) = N_T(6/30^{4/3})D^{1+\frac{1}{3}}$
Coefficient of locality $\sim \sqrt[3]{D}$.

mxw3D = accelerator (GPGPU) + LRnLA algorithms

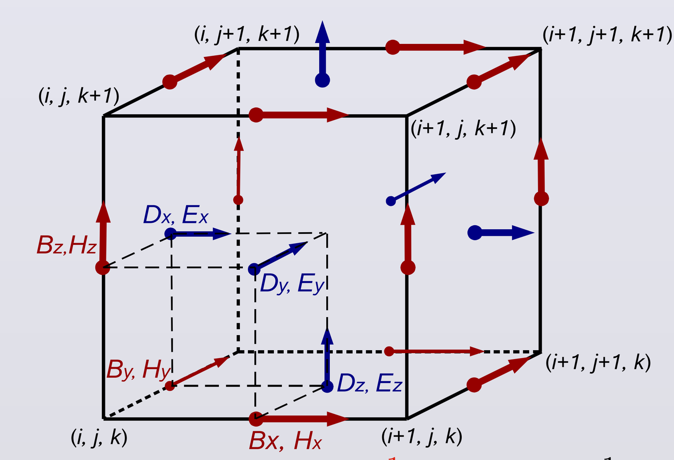
$$\frac{1}{c}\frac{\partial \vec{B}}{\partial t} = -\nabla \times \vec{E}, \quad \mu\vec{H} = \vec{B}, \quad \frac{1}{c}\frac{\partial \vec{D}}{\partial t} = \nabla \times \vec{H}, \quad \varepsilon\vec{E} = \vec{D}$$

FDTD scheme for Maxwell's equations:

$s = 1, 2, 3,$
$p = (s+1)\%3,$
$m = (s-1)\%3$

$\hat{\Delta}_s$ is finite operator for (2^{nd}, 4^{th} orders)

$\hat{\mu}_s^{-1}, \hat{\varepsilon}_s^{-1}$ operators for media model.



$$B_{s,i}^{k+\frac{1}{2}} = B_{s,i}^{k-\frac{1}{2}} - c\Delta t\left(\hat{\Delta}_p E_{m,\tilde{i}}^{k} - \hat{\Delta}_m E_{p,\tilde{i}}^{k}\right), \quad H_{s,i}^{k+\frac{1}{2}} = \hat{\mu}_s^{-1}\vec{B}_i^{k+\frac{1}{2}},$$

$$D_{s,\tilde{i}}^{k+1} = D_{s,\tilde{i}}^{k} + c\Delta t\left(\hat{\Delta}_p H_{m,i}^{k+\frac{1}{2}} - \hat{\Delta}_m H_{p,i}^{k+\frac{1}{2}}\right), \quad E_{s,\tilde{i}}^{k+1} = \hat{\varepsilon}_s^{-1}\vec{D}_{\tilde{i}}^{k+1},$$

	+/-	fma	ld	st	+/-	fma	ld	st
$\hat{\Delta}_s$	1	1	3	1	2	2	6	1
$\hat{\mu}_s^{-1}, \hat{\varepsilon}_s^{-1}$	0	1	1	1	4	4	11	1
6 pair	12	18	42	6	48	48	72	12
cells/sec	$80 \cdot 10^9$		$1.5 \cdot 10^9$		$25 \cdot 10^9$		$0.8 \cdot 10^9$	

peak performance GeForce Titan: 5TFlops (float),
bandwidth: 288.4GB/sec

Pop-up problems during the realization of algorithms on architecture Kepler

- Caches (L1 and L2) are too small for data localization in numerical 3D simulation, to avoid this problem the register file of SM(X) module (256KB per module) is suitable to use;
- Increasing the locality leads to limit of the maximum of vector length; it is optimal to consider quasi-2D tasks with grid size ratio as $Nx > Ny > Nz = 64 \div 256$.

Specifics of CUDA realization of LRnLA algorithms

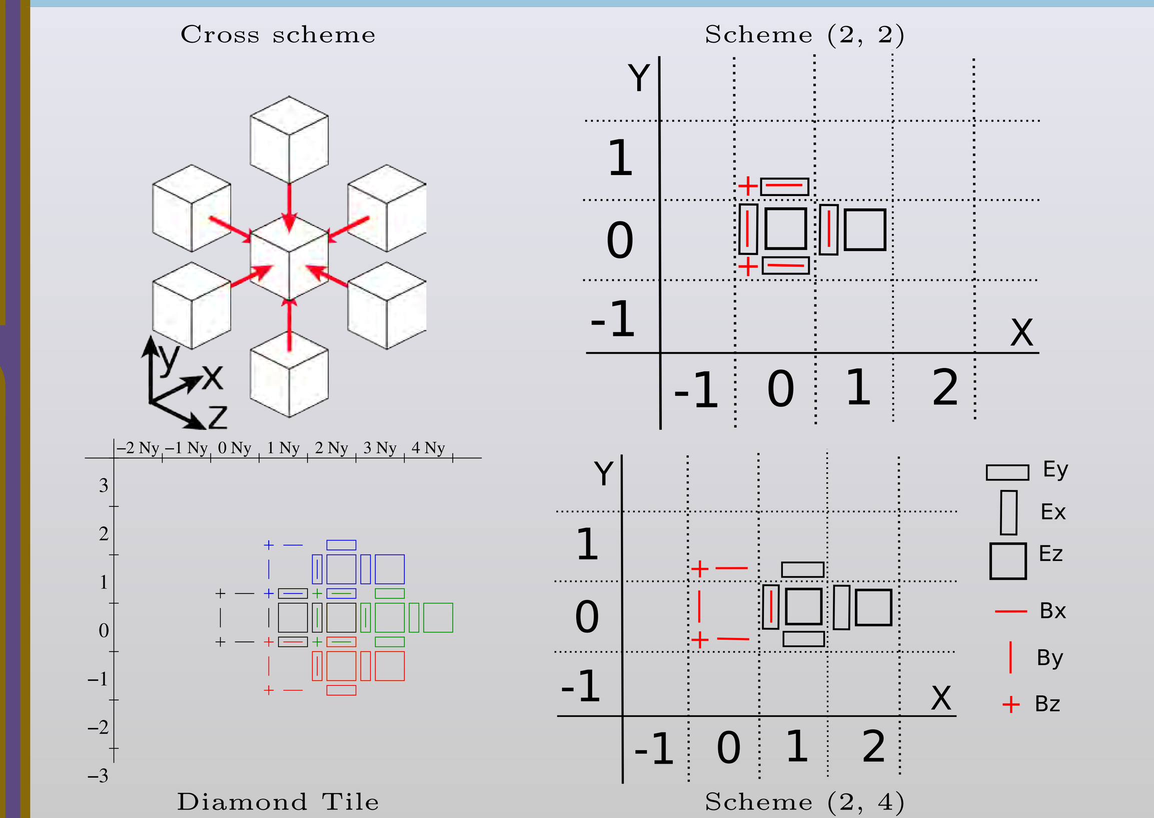
- CUDA-thread index corresponds to Yee Cell index along Z-axis for solving the problems with vectorization, that is mean aligned access to memory, code homogeneity, occupancy of SMX module cores. Shared memory are explored for execution of finite-difference operators $\partial/\partial z$ along Z-axis.
- CUDA-block index corresponds to Yee Cell index along Y-axis to provide asynchronous execution of calculations by different SM-modules without data dependences (it is odd and even passes required).
- CUDA-kernel consist of N_t loop steps which are executed along $x - \tilde{c}t$ axis. At each step electromagnetic field components of Yee Cells trapped to diamond region are executed. Diamonds splits two types for electric and magnetic fields and are named Diamond Domino. The distance between halves of Domino along X-axis is quoted by numerical scheme order. The size of diamond region is $2 \cdot N_{\mathrm{DT}} \times N_{\mathrm{DT}}$ Yee Cells chosen for increasing the locality of calculations. N_{DT} is varied.
- Host calls CUDA-kernel N_x/N_{DT} times in the loop when moves along X-axis. Before the end of the loop all data are moved to N_t steps and corresponding components of the field may be visualised or saved for further usage and analysis.

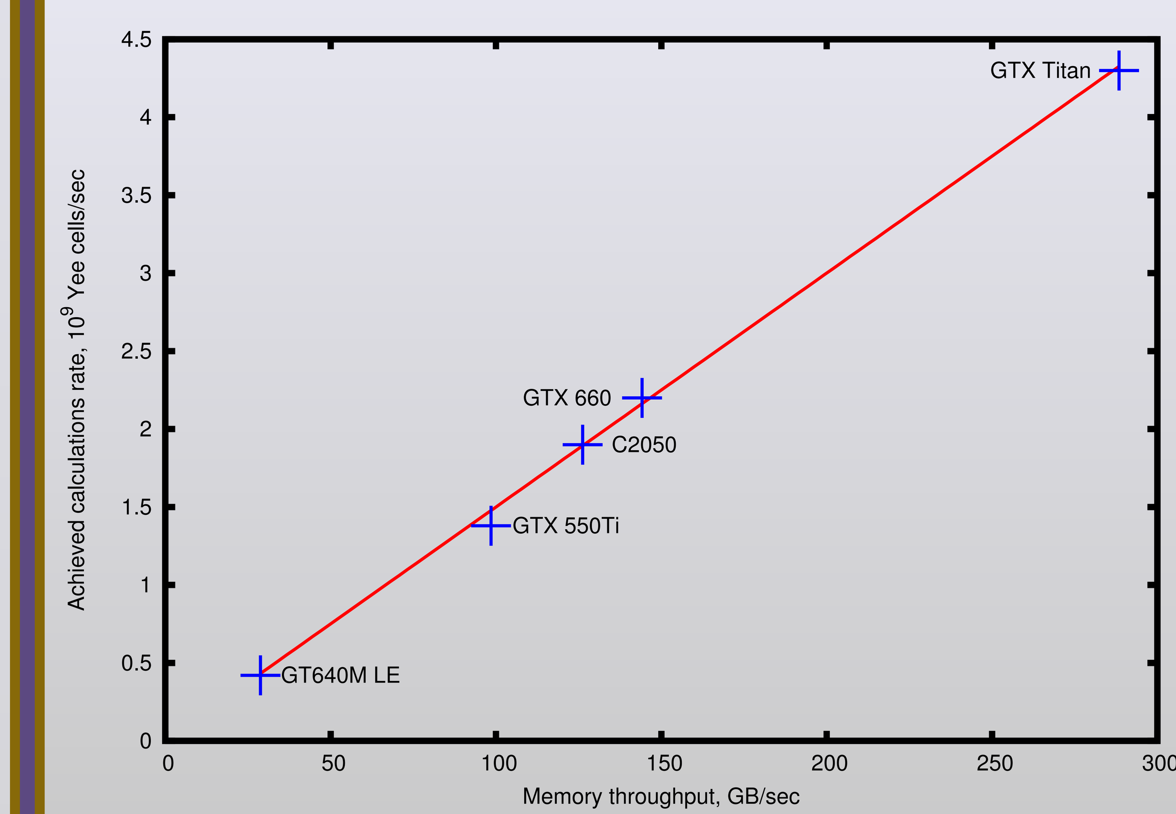
The table below discribes agreement between hierarchy levels of memory system and parallelism for Diamond Tile algorithms.

	SIMT	DLP	ILP	TLP	PLP
registers	i_z=tid.x		$4n^3$ Cell/DT → fetches of independent calculations		
share	$\frac{\partial}{\partial z}$	aligned coalesced exchange per 128B			
cache L1					
cache L2				DTs along y is asynch.	
GDDR5					window of calculations
PCIe					
DDR3/4					
	thread N_z	warp $N_z/32$	block $N_y/2n$	grid Nx/n	UVM

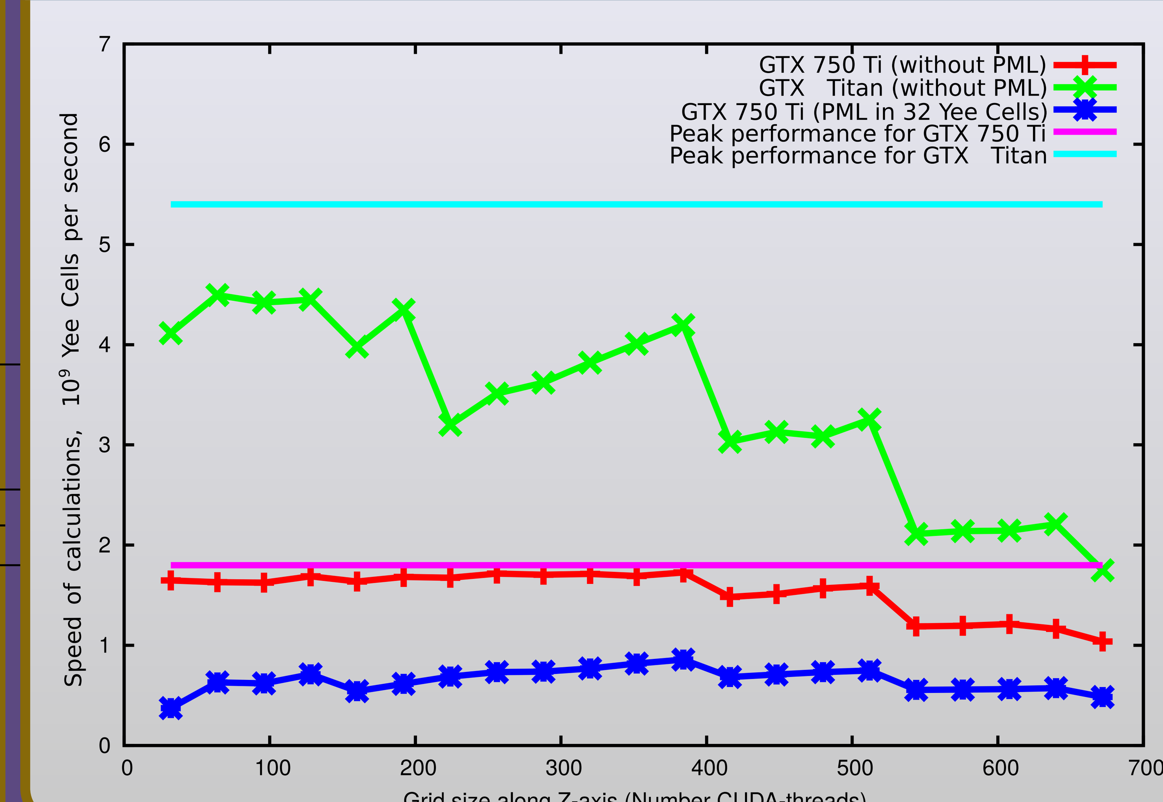
where
$N_x \times N_y \times N_z$ Yee Cells is simulation domain;
n is linear size of Diamond Tile (handles $2n^2$ Yee Cells in $2n$ time layer)
$\frac{\partial}{\partial z}$ access to neibour Cells along z axis accoding cross scheme ($i_z \pm 1, \dots$)

Diamond Domino and Diamond Tile algorithms





Result efficiency



Conclusion

1. Model of calculation is created and developed for heterogeniuos systems involving GPGPU.
2. Model of calculation is implemented for the problem of full matrix multiplication.
3. New class LRnLA algoruthms was developed and named Diamond Domino/Diamond Tile. These algorithms are optimized in relation of calculation locality for explicit finite-difference methods with local cross scheme.
4. Developed algorithms give the performance around 33% from peak and operate with 90% communication resources of memory hierarchy. The speed of calculations achieve 10^9 Yee Cells per second for Maxwell architecture (750 Ti) and $5 \cdot 10^9$ Yee Cells per second for Kepler architecture (GTX Titan Black).

Bibliography

1. Kurzak J., Tomov S., Dongarra J., «Autotuning GEMMs for Fermi», SC11 November 12-18, 2011, Seattle, Washington, USA.
2. Lebesgue H., «Lecons sur l'Integration et la Recherche des Fonctions Primitives», Gauthier-Villars, 1904.
3. http:/www.kiam.ru/MVS/resourses/k100.html
4. Levchenko V.D. «Asynchronous algorithms as the method of reaching by the effectiveness of calculations. Information technologies and computing systems», 2005 (1).
5. J.Sanders, E.Kandrot, «CUDA by Example: An Introduction to General-Purpose GPU Programming».
6. Kane Yee, «Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media», IEEE Transactions on Antennas and Propagation 14 (3): 302–307, 1966.