Parallel programming technologies on hybrid architectures

Streltsova O.I., Podgainy D.V.

Laboratory of Information Technologies

Joint Institute for Nuclear Research

SCHOOL ON JINR/CERN GRID
AND ADVANCED INFORMATION SYSTEMS
Dubna, Russia
23, October 2014

Goal: Efficient parallelization of complex numerical problems in computational physics

Plan of the talk:

I. Efficient parallelization of complex numerical problems in computational physics
- Introduction
- Hardware and software
- Heat transfer problem

II. GIMM FPEIP package and MCTDHB package

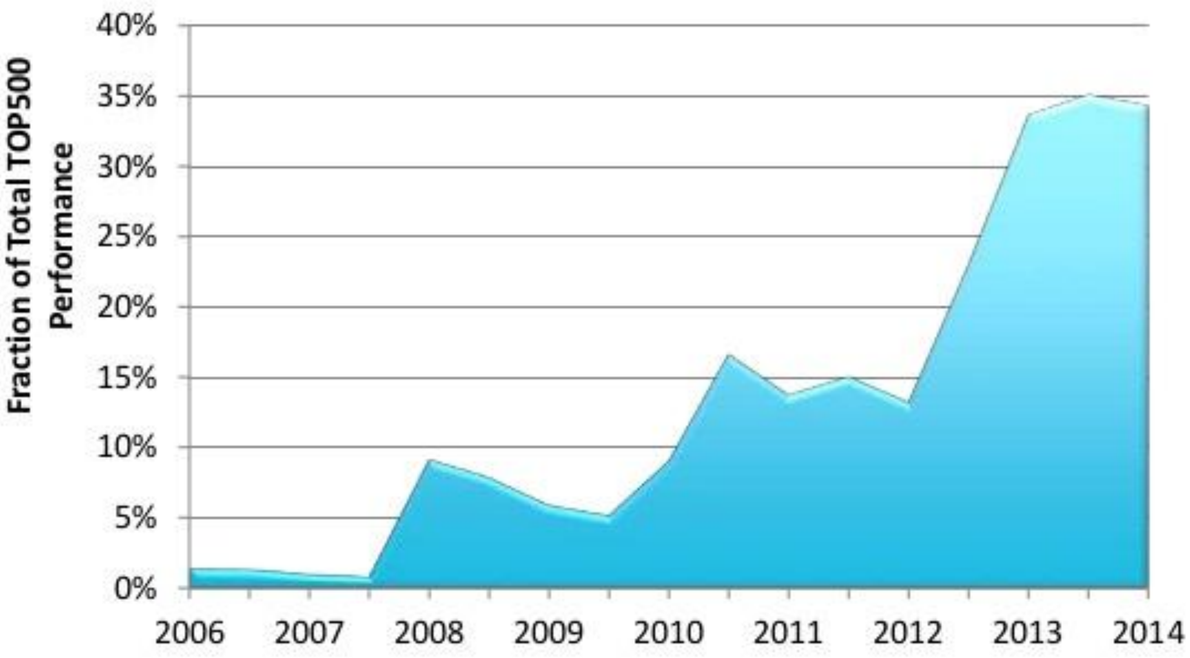
III. Summary and conclusion

TOP500 List – June 2014

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
		■ ■ ■				
41	Barcelona Supercomputing Center Spain	**MareNostrum** - iDataPlex DX360M4, Xeon E5-2670 8C 2.600GHz, Infiniband FDR IBM	48896	925.1	1017.0	1016
42	Moscow State University - Research Computing Center Russia	**Lomonosov** - T-Platforms T-Blade2/1.1, Xeon X5570/X5670/E5630 2.93/2.53 GHz, Nvidia 2070 GPU, PowerXCell 8i Infiniband QDR T-Platforms	78660	901.9	1700.2	2800
43	Rensselaer Polytechnic Institute United States	**AMOS** - BlueGene/Q, Power BQC 16C 1.6GHz, Custom Interconnect IBM	81920	894.4	1048.6	411

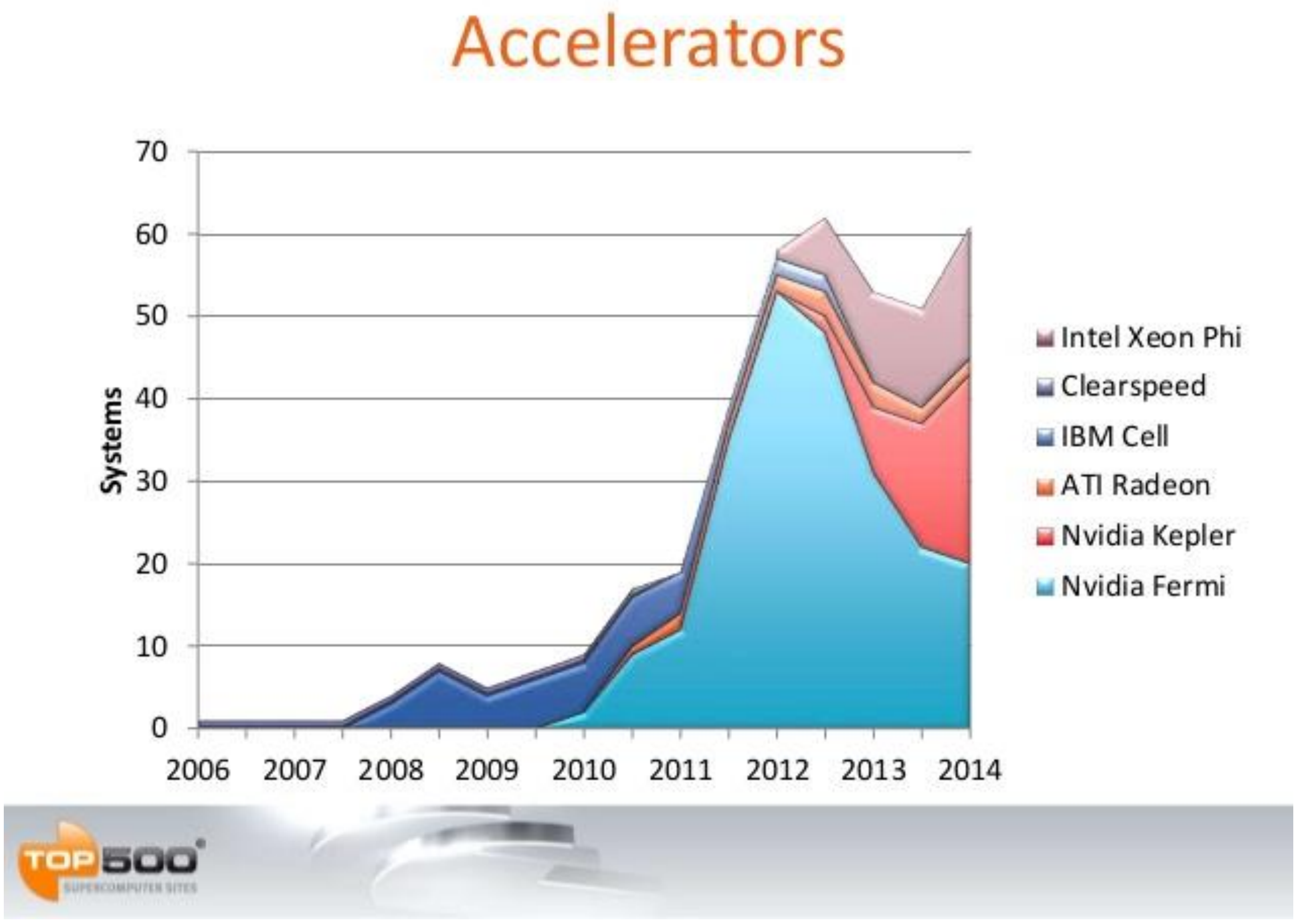
TOP500 List – June 2014

Performance Share of Accelerators



Source:
http://www.top500.org/blog/slides-for-the-43rd-top500-list-now-available/

TOP500 List – June 2014



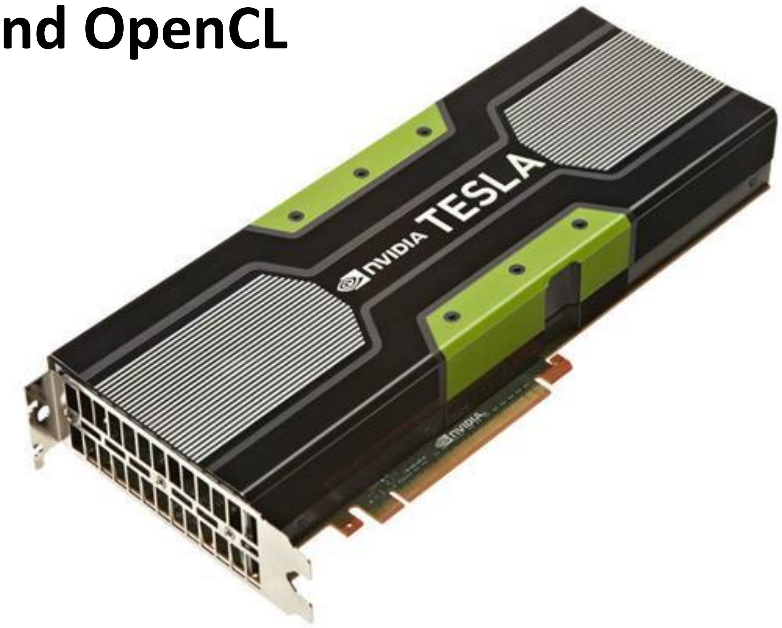
Source:
http://www.top500.org/blog/slides-for-the-43rd-top500-list-now-available/

«Lomonosov» Supercomputer , MSU

>**5000** computation nodes
Intel Xeon X5670/X5570/E5630, PowerXCell 8i
~36 Gb DRAM
2 x **nVidia Tesla** X2070 6 Gb GDDR5 (448 CUDA-cores)
InfiniBand QDR

NVIDIA Tesla K40 “Atlas” GPU Accelerator

- **Custom languages such as CUDA and OpenCL**
- **Specifications**



- **2880** CUDA GPU cores
- **Peak precision floating point performance**

  4.29 TFLOPS single-precision

  1.43 TFLOPS double-precision
- **memory**

  12 GB GDDR5

  Memory bandwidth up to **288** GB/s

Supports Dynamic Parallelism and HyperQ features

«Tornado SUSU» Supercomputer,
South Ural State University, Russia

«Tornado SUSU» supercomputer took the [157 place in 43-th issue of TOP500 rating (June 2014)].

480 computing units (compact and powerful computing blade-modules)

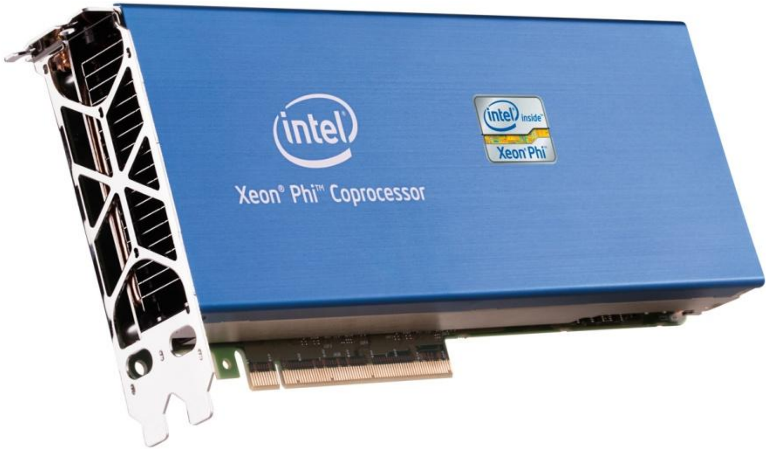
960 processors Intel Xeon X5680

(Gulftown, 6 cores with frequency 3.33 GHz)

384 coprocessors Intel Xeon Phi SE10X (61 cores with frequency 1.1 GHz)

Intel® Xeon Phi™ Coprocessor

Intel Many Integrated Core Architecture (Intel **MIC**) is a multiprocessor computer architecture developed by Intel.

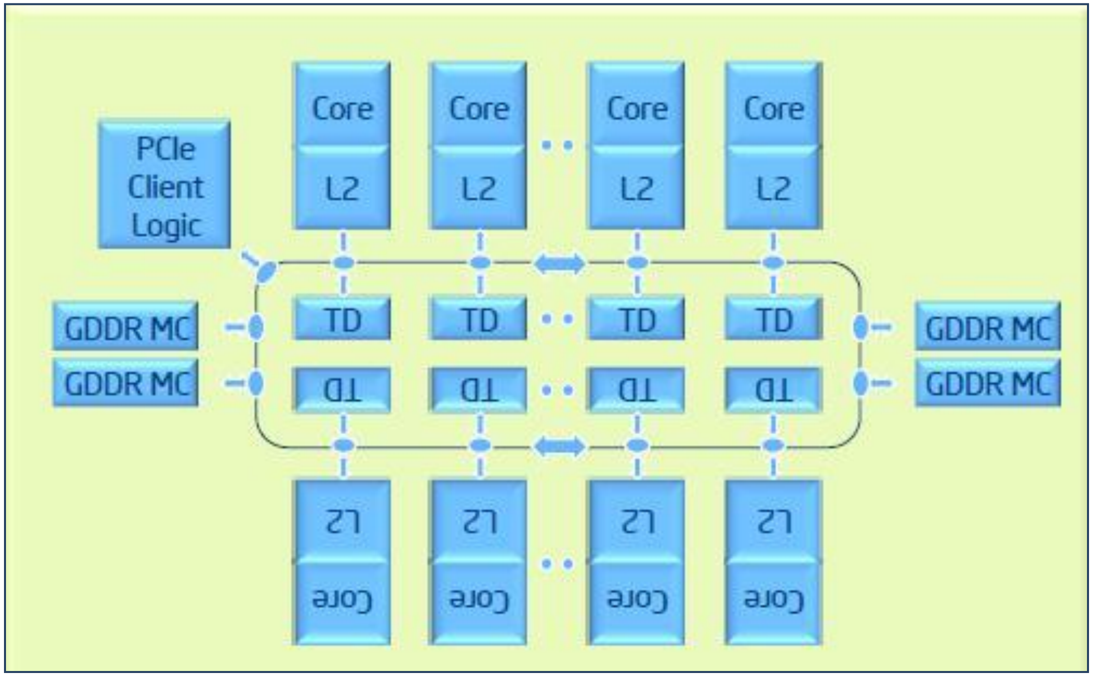


At the end of 2012, Intel launched the first generation of the Intel Xeon Phi product family.

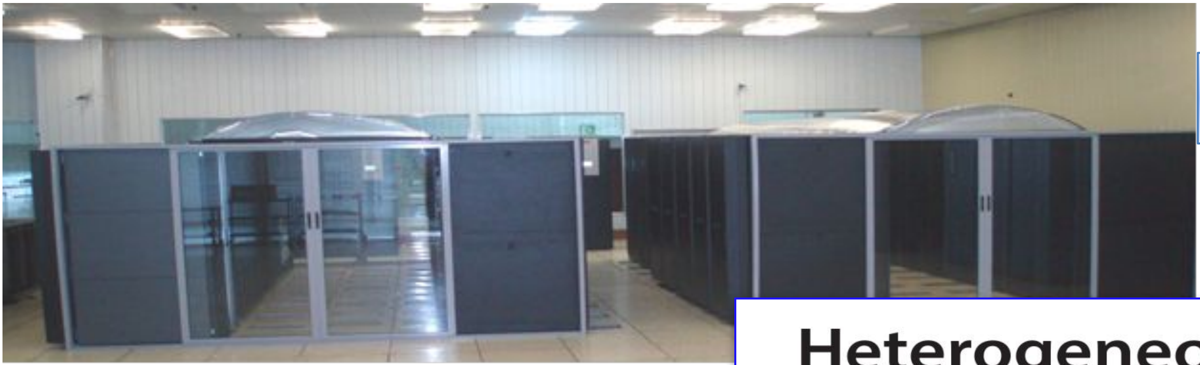
Intel Xeon Phi 7120P

Clock Speed	**1.24 GHz**
L2 Cache	**30.5 MB**
TDP	**300 W**
Cores	**61**
More threads	**244**

The core is capable of supporting **4 threads** in hardware.



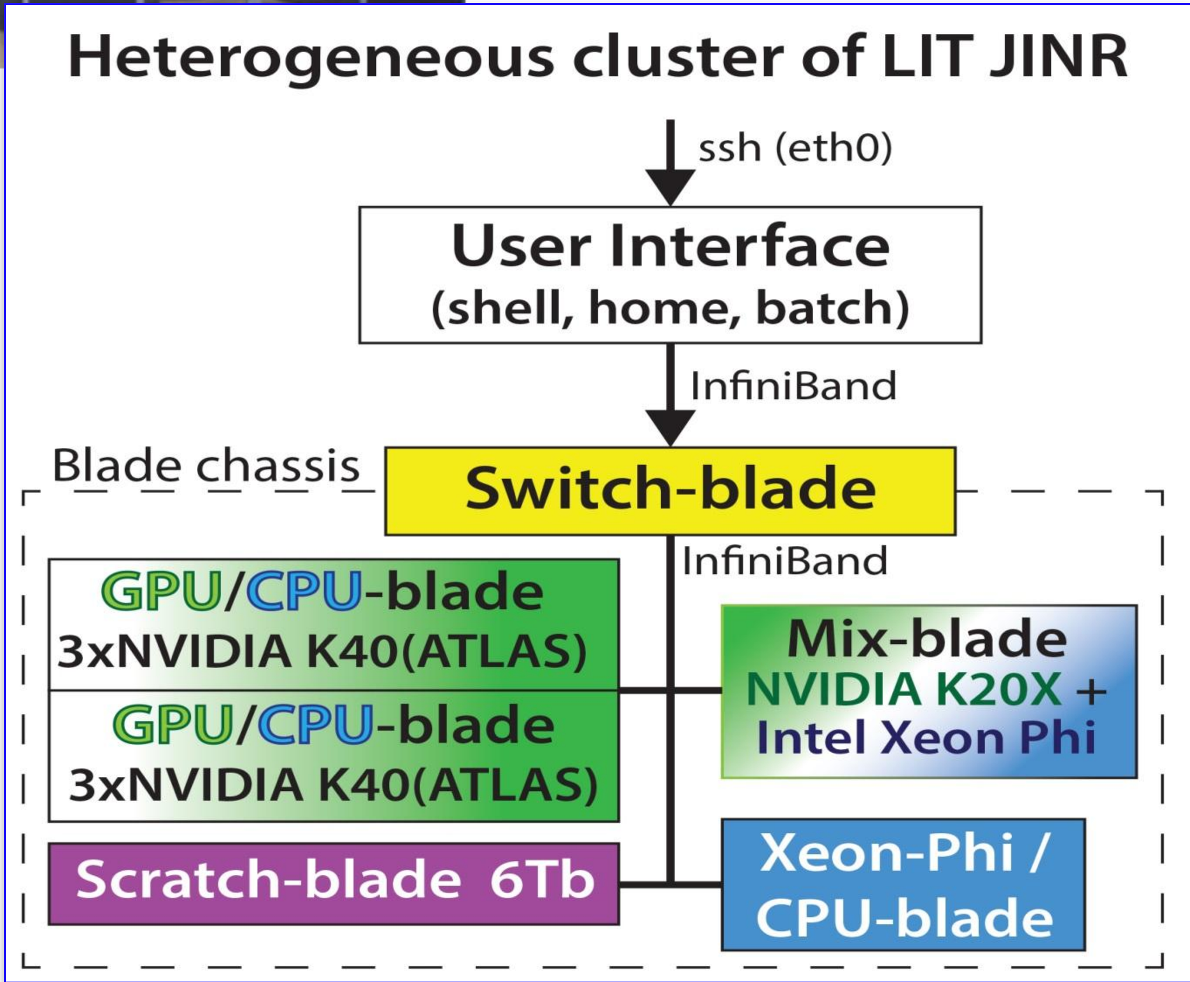
HybriLIT: heterogeneous computation cluster



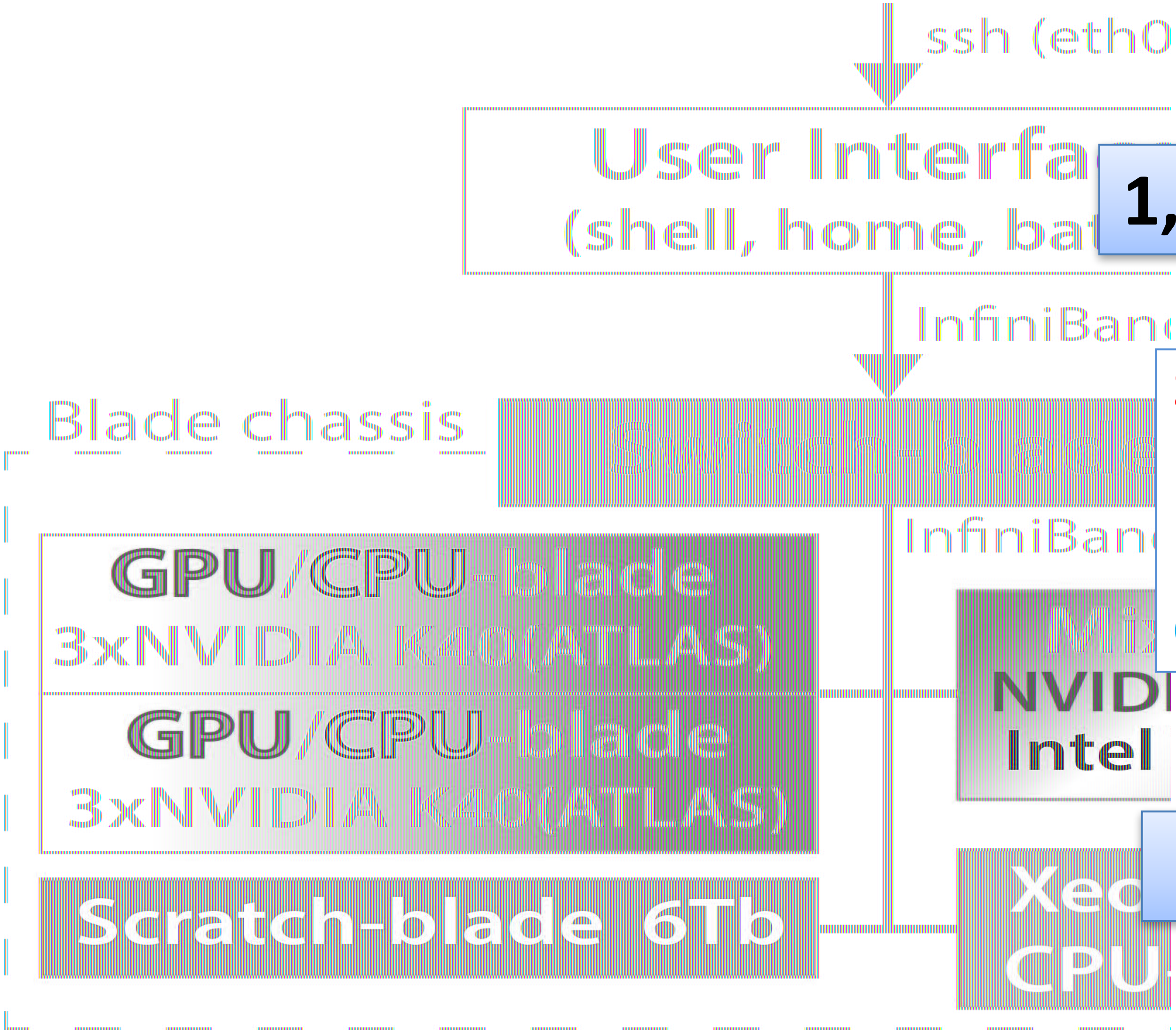
Site: **http:// hybrilit.jinr.ru**

CICC comprises
2582 Cores
Disk storage capacity
1800 TB

August, 2014



HybriLIT: heterogeneous computation cluster



1,2

2x Intel Xeon CPU E5-2695v2
3x NVIDIA TESLA K40S

3

2x Intel Xeon CPU E5-2695v2
NVIDIA TESLA K20X
Intel Xeon Phi Coprocessor 5110P

4

2x Intel Xeon CPU E5-2695v2
2x Intel Xeon Phi Coprocessor 7120P

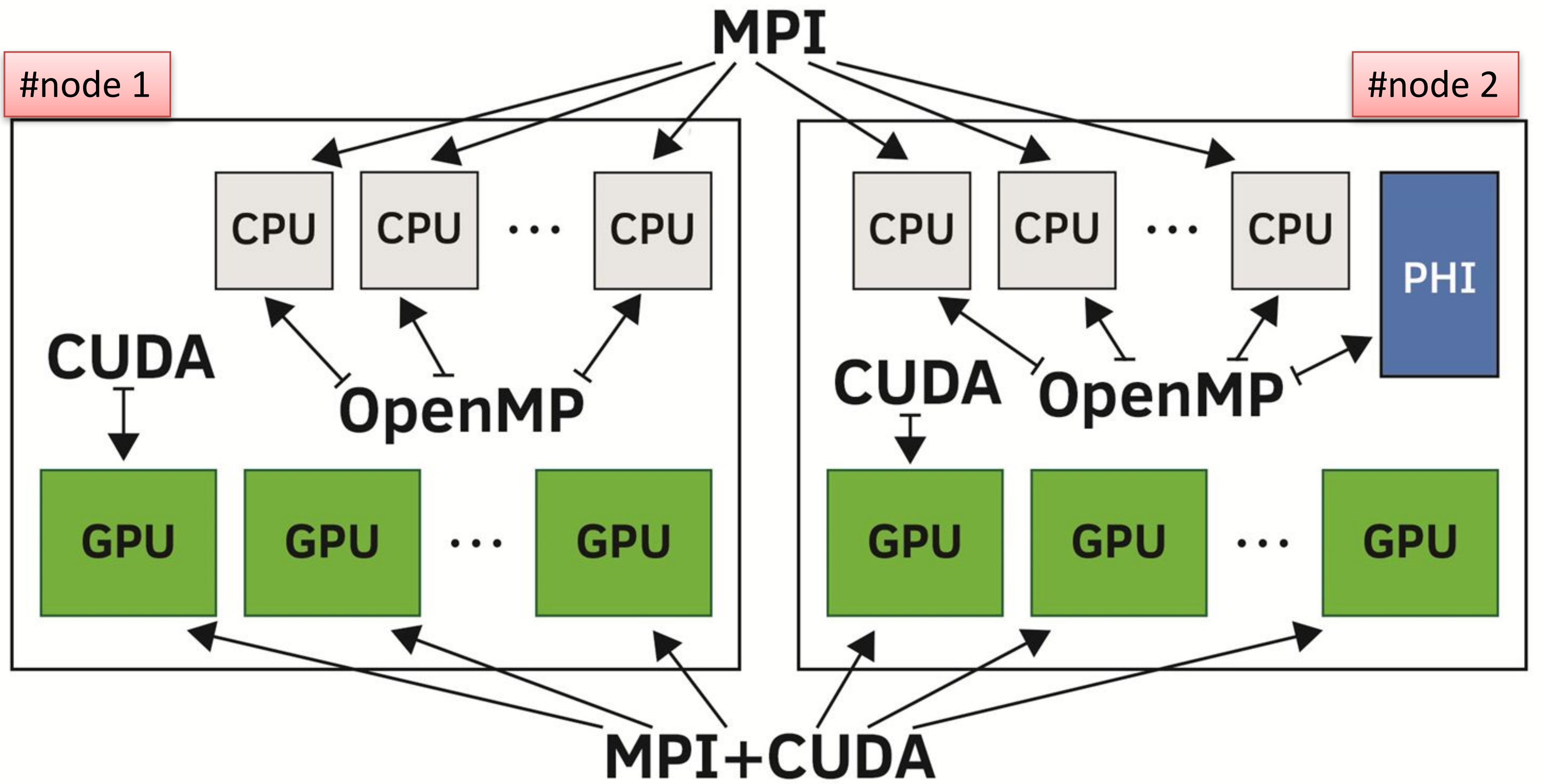
What we see: modern Supercomputers are hybrid with heterogeneous nodes

- Multiple CPU cores with share memory
- Multiple GPU

- Multiple CPU cores with share memory
- Multiple Coprocessor

- Multiple CPU
- GPU
- Coprocessor

Parallel technologies: levels of parallelism



How to control hybrid hardware:
MPI – OpenMP – CUDA - OpenCL ...

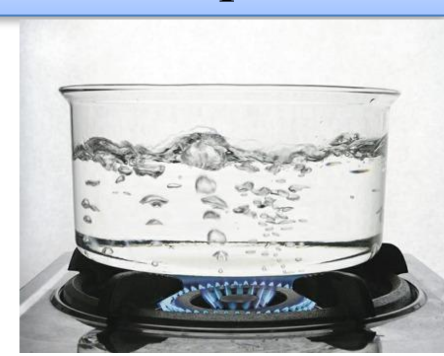
In the last decade novel computational facilities and technologies has become available: MPI-OpenMP-CUDA-OpenCL...

It is not easy to follow modern trends.
Modification of the existing codes or developments of new ones ?

Problem HCE: heat conduction equation

Initial boundary value problem for the heat conduction equation:

$$\begin{cases} \dfrac{\partial u}{\partial t} = Lu + f(x,y,t), \ (x,y) \in D, \ t > 0; \\ u\big|_{t=0} = u_0(x,y), \ (x,y) \in \overline{D}; \ u\big|_{\Gamma} = \mu(x,y,t), \ t \geq 0, \end{cases}$$



- D – rectangular domain with boundary Γ:

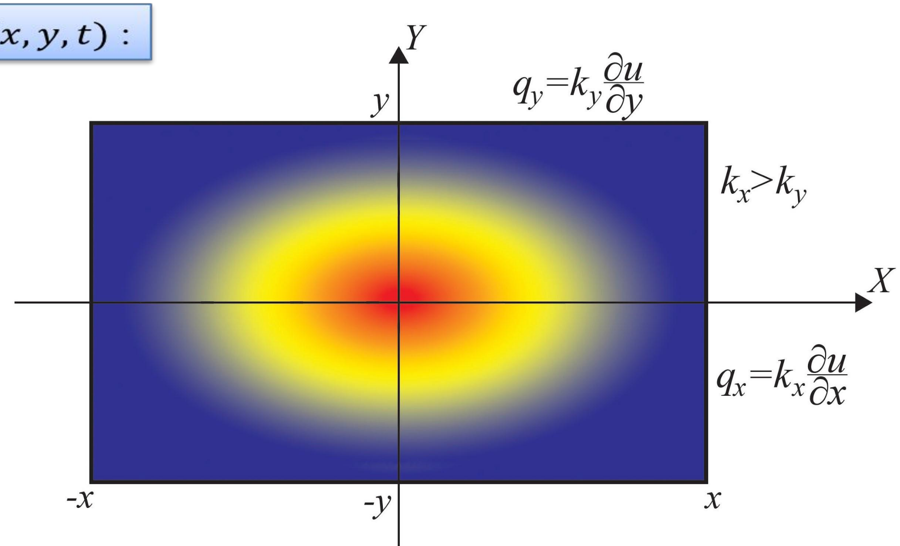
$$\overline{D} = D + \Gamma = \{(x,y): x_L \leq x \leq x_R, \ y_L \leq y \leq y_R\}$$

- L is a linear differential operator acting on $u(x,y,t)$:

$$L = L_1 + L_2,$$

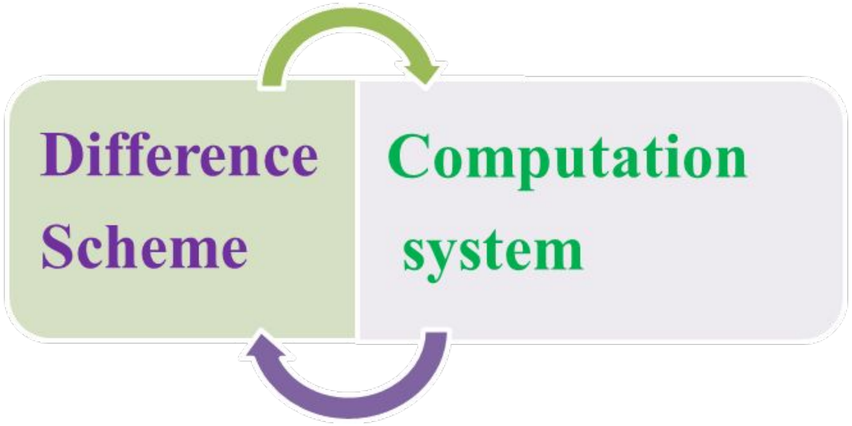
$$L_1 u = \frac{\partial}{\partial x} K_1(x,y,t) \frac{\partial u}{\partial x},$$

$$L_2 u = \frac{\partial}{\partial y} K_2(x,y,t) \frac{\partial u}{\partial y}.$$



Problem HCE: computation scheme

Difference scheme:
Explicit, implicit, … ?



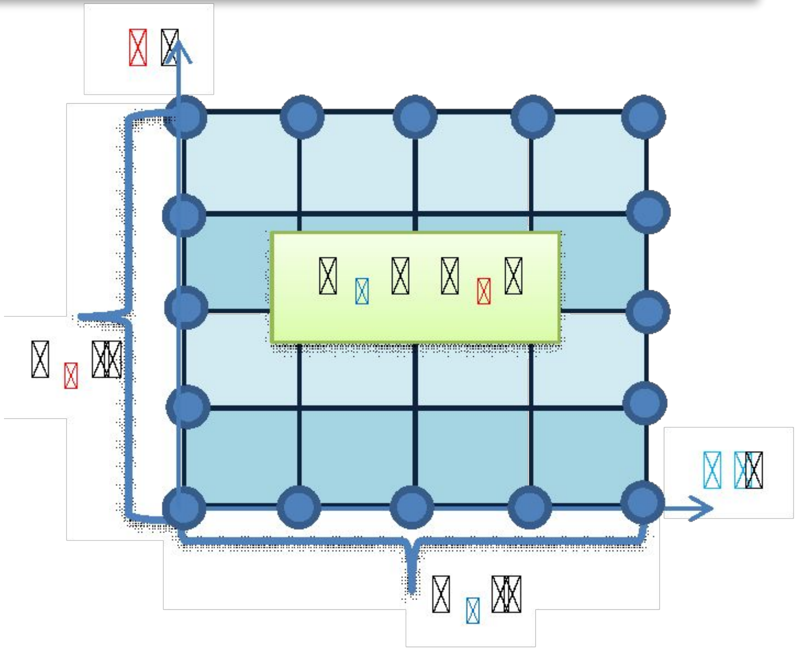
Locally one-dimensional scheme:
reduction of a multidimensional problem to a chain of one-dimensional problems

Let: $\overline{\omega} = \overline{\omega}_\tau \times \omega_{h_x h_y}$:

$$\omega_{h_x h_y} = \overline{\omega}_{h_x} \times \overline{\omega}_{h_y}, \overline{\omega}_\tau = \left\{ t_j = j\tau,\ j = \overline{0, N_t - 1} \right\},$$

$$\overline{\omega}_{h_x} = \left\{ x_{i_1} = x_L + i_1 h_x,\ i_1 = \overline{0, N_x - 1} \right\},$$

$$\overline{\omega}_{h_y} = \left\{ y_{i_2} = y_L + i_2 h_y,\ i_2 = \overline{0, N_y - 1} \right\}$$



- L is a linear differential operator acting on $u(x, y, t)$:

Problem HCE: computation scheme

Step 1:
Difference equations *(Ny-2)* on *x* direction

$$\frac{v_{(1)}^{j+1}-v_{(2)}^{j}}{\tau}=\Lambda_1 v_{(1)}^{j+1}+\varphi_1,\ \ \Lambda_1 v=\left(a_1 v_{\bar{x}}\right)_x,\ \ a_1=K_1\left(x_{i_1-\frac{1}{2}},y_{i_2},t\right),$$

Step 2:
Difference equations *(Nx-2)* on *y* direction

$$\frac{v_{(2)}^{j+1}-v_{(1)}^{j}}{\tau}=\Lambda_2 v_{(2)}^{j+1}+\varphi_2,\ \ \Lambda_2 v=\left(a_2 v_{\bar{y}}\right)_y,\ \ a_2=K_2\left(x_{i_1},y_{i_2-\frac{1}{2}},t\right),$$

$$v_{(\alpha)}^{j}=v_{(\alpha)}\left(x_{i_1},y_{i_2},t_j\right),\ \ \alpha=1,2;\ \ \left(x,y,t\right)\in\omega;\ \ x_{i_1-\frac{1}{2}}=x_{i_1}-\frac{1}{2}h_x,\ \ y_{i_2-\frac{1}{2}}=y_{i_2}-\frac{1}{2}h_y.$$

under the additional conditions of conjugation, boundary conditions and normalization condition

$$v_{(2)}\left(x,y,t_j\right)=v_{(1)}\left(x,y,t_{j+1}\right),\ \ j=\overline{0,N_t-2},$$

$$v_{(2)}\left(x,y,0\right)=u_0\left(x,y\right),\ \ \left(x,y\right)\in\bar{\omega}_{h_x h_y};$$

$$v_{(\alpha)}^{j}=\mu(x,y,t_j),\ \ \alpha=1,2,(x,y)\in\gamma^a;$$

$$\varphi_1+\varphi_2=f$$

Problem HCE: parallelization scheme

- L is a linear differential operator acting on $u(x, y, t)$:

Parallel

- L is a linear differential operator acting on $u(x, y, t)$:

Parallel

Parallel Technologies

CPU

- OpenMP **realization**
- (Multiple **CPU** cores that share memory)
- Optimizations

Xeon Phi

- OpenMP **realization for Coprosseror**
- Optimizations

GPU

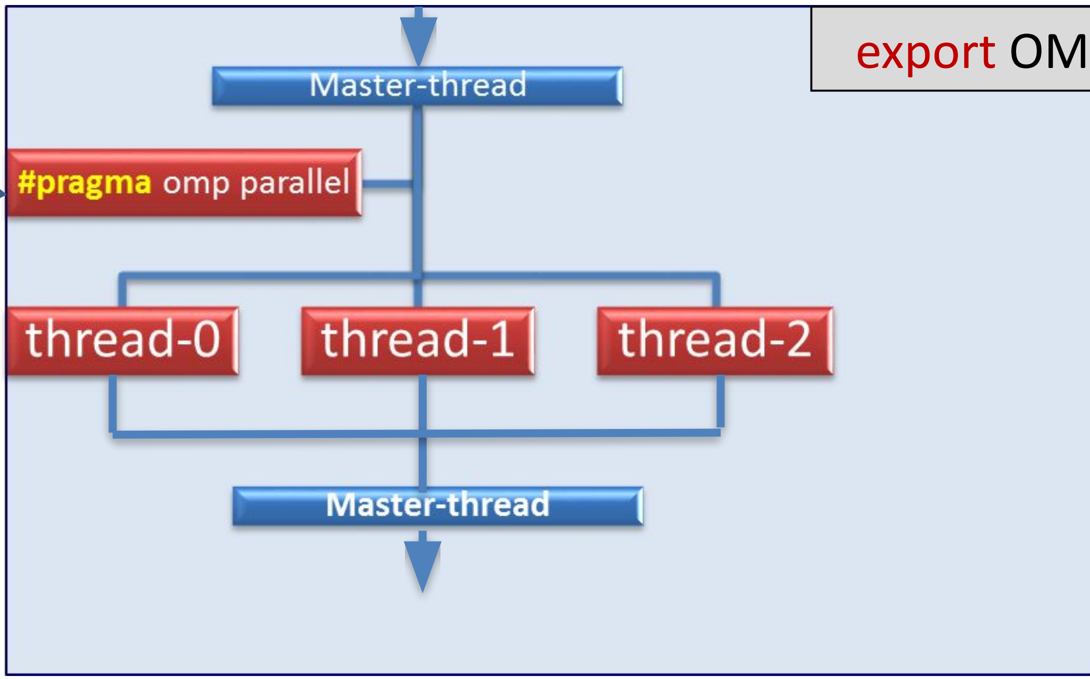
- CUDA **realization**
- Optimizations

OpenMP realization of parallel algorithm

OpenMP (Open specifications for Multi-Processing)

OpenMP (**Open specifications for M**ulti-**P**rocessing) is an API that supports multi-platform shared memory multiprocessing programming in **Fortran, C, C++.**

- Compiler directives
- Library routines
- Environment variables



export OMP_NUM_THREADS=**3**

http://openmp.org/wp/

OpenMP (Open specifications for Multi-Processing)

```
#include <stdio.h>
#include <omp.h>

int main (int argc, char *argv[]) {
    const int N = 1000;
    int  i, nthreads;
    double  A[N];
     nthreads = omp_get_num_threads();
    printf("Number of thread = %d \n, nthreads);

 #pragma omp parallel for
 for (i = 0; i < N; i++) {
      A[i] = function(i);
  }
  return 0;
}
```

Library routines

Compiler directive

Use flag **-openmp** to compile using Intel compilers:

icc –openmp code.c –o code

OpenMP realization:
Multiple CPU cores that share memory

Table 2. OpenMP realization problem 1:
execution time and acceleration (CPU Xeon *K100 KIAM RAS*)

Number of threads	Time (sec)	Acceleration
1	84.64439	1
2	46.93157	1,80357
4	23.46677	3,60699
6	17.19202	4,92347
8	14.08791	6,0083
10	12.47396	**6,78569**

OpenMP realization: Intel® Xeon Phi™ Coprocessor

Compiling:

```
icc  -openmp -O3  -vec-report=3 -mmic algLocal_openmp.cc –o
alg_openmp_xphi
```

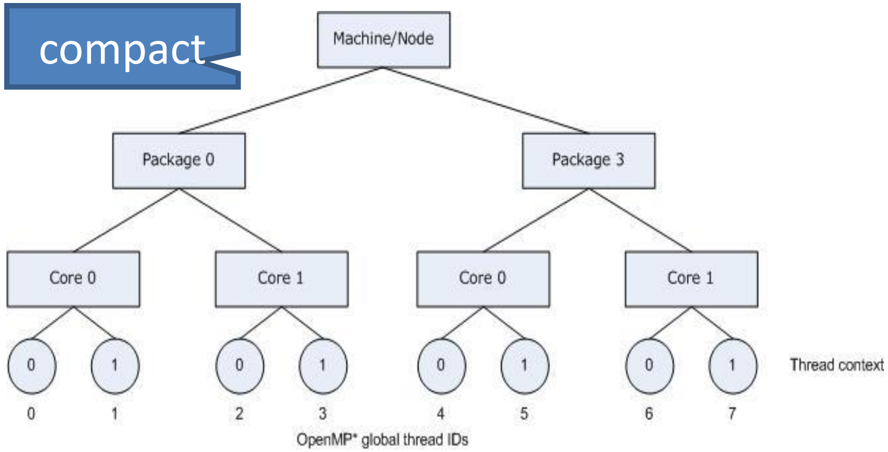
Table 3. OpenMP realization: Execution time and Acceleration (Intel Xeon Phi, LIT).

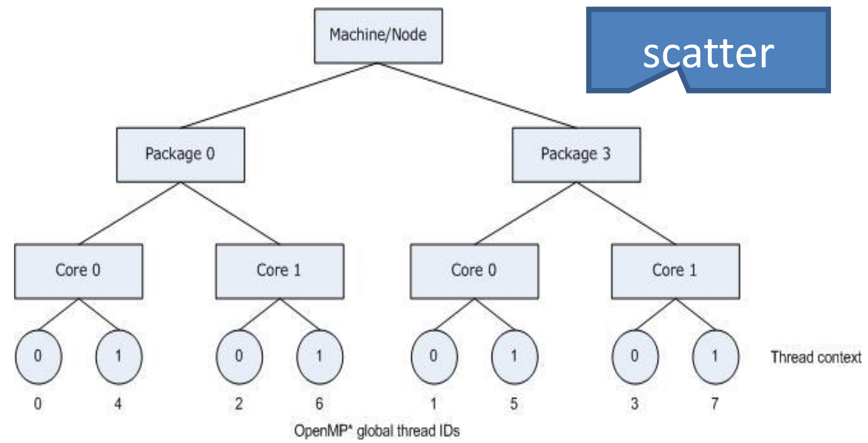
$N_x \times N_y$	Time CPU [sec]	Time Xeon Phi [sec]	Acceleration
500×500	33.976	4.489	7,568
1000×1000	143.693	14.632	9,82
2000×2000	595.058	54.229	**10,973**
3000×3000	1349.978	123.998	10,887
4000×4000	2406.355	229.664)	10,477

OpenMP realization: Intel® Xeon Phi™ Coprocessor Optimizations

The **KMP_AFFINITY** Environment Variable: The Intel® OpenMP* runtime library has the ability to bind OpenMP threads to physical processing units.
The interface is controlled using the **KMP_AFFINITY** environment variable.

compact





scatter

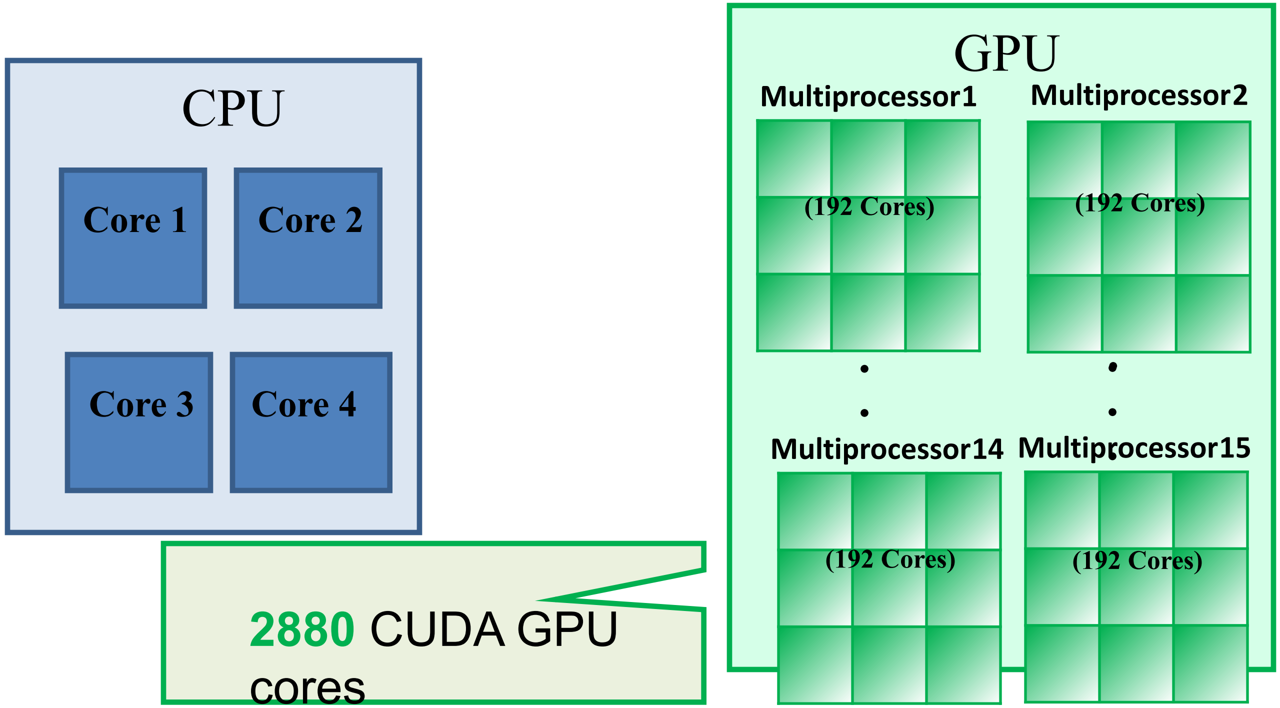
KMP_AFFINITY	compact	scatter	balanced
Execution time (sec)	10.85077	13.24356	10.46549

Source: https://software.intel.com/

CUDA (Compute Unified Device Architecture) programming model, CUDA C

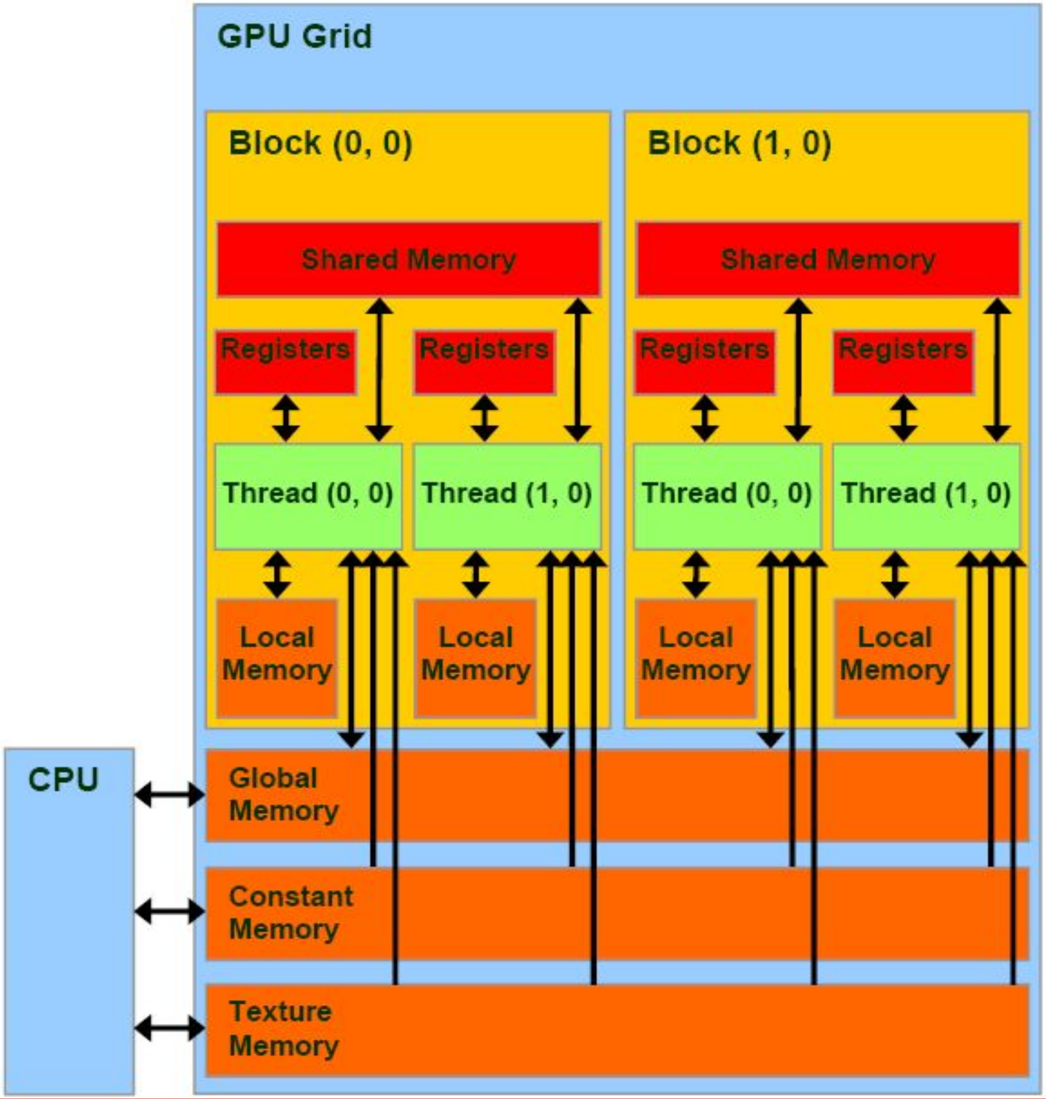
CUDA (Compute Unified Device Architecture) programming model, CUDA C

CPU / GPU Architecture



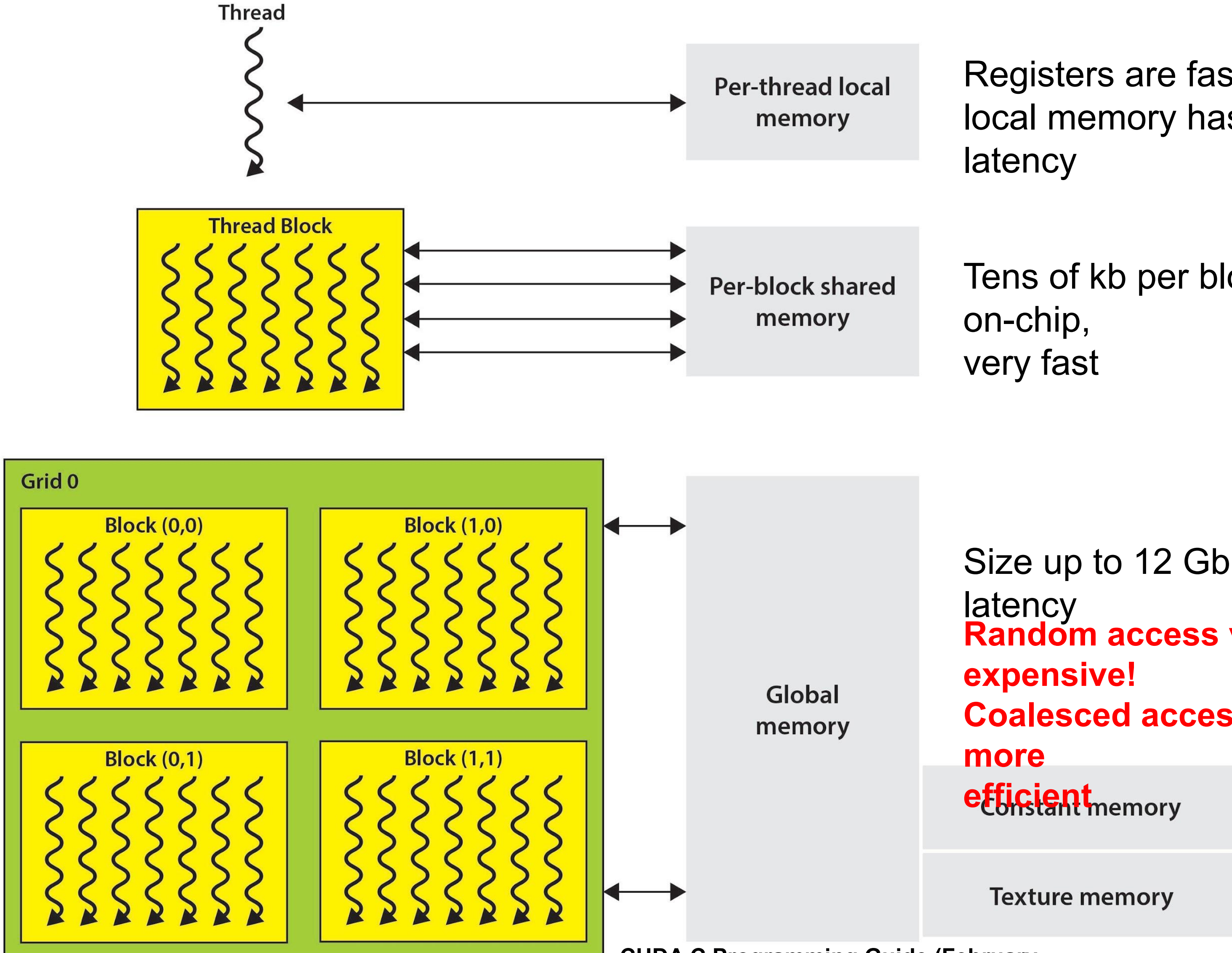
Source:
http://blog.goldenhelix.com/?p=374

CUDA (Compute Unified Device Architecture) programming model



Source:
http://www.realworldtech.com/includes/images/articles/g100-2.gif

Device Memory Hierarchy



Registers are fast, off-chip local memory has high latency

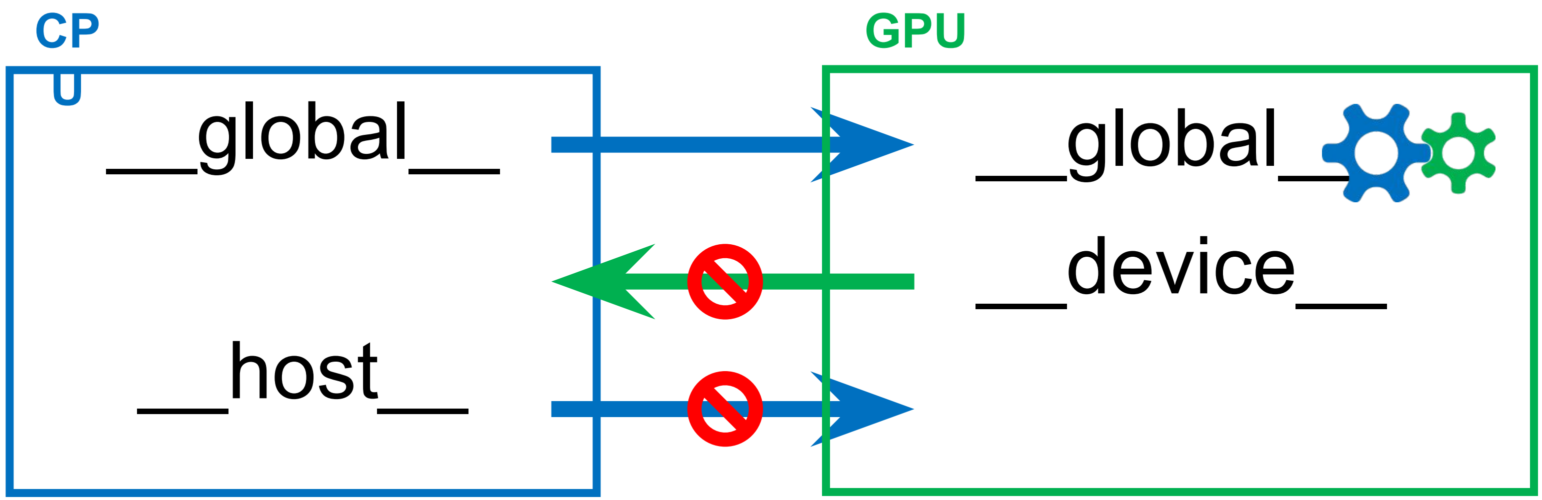
Tens of kb per block, on-chip, very fast

Size up to 12 Gb, high latency

Random access very expensive!
Coalesced access much more efficient

CUDA C Programming Guide (February

Function Type Qualifiers




```
__global__ void kernel ( void ){
}

int main{
   …
   kernel <<< gridDim, blockDim >>> ( args );
   …
}
```

Language extensions:
Kernel execution directive

dim3 **gridDim** – dimension of grid,
dim3 **blockDim** – dimension of blocks

Threads and blocks

Block 0	**Thread 0**	**Thread 1**
Block 1	Thread 0	Thread 1
Block 2	Thread 0	Thread 1
Block 3	Thread 0	Thread 1

tid – index of threads

```
int tid = threadIdx.x + blockIdx.x * blockDim.x
```

Scheme program on CUDA C/C++ and C/C++

CUDA	C / C++
1. Memory allocation	
cudaMalloc (void ** devPtr, size_t size);	void * malloc (size_t size);
2. Copy variables	
cudaMemcpy (void * dst, const void * src, size_t count, enum cudaMemcpyKind kind);	void * memcpy (void * destination, const void * source, size_t num);
copy: host ⟶ device, device ⟶ host, host ⟷ host, device ⟷ device	---
3. Function call	
kernel <<< gridDim, blockDim >>> (args);	double * Function (args);
4. Copy results to host	
cudaMemcpy (void * dst, const void * src, size_t count, device → host);	---

Compilation

Compilation tools are a part of CUDA SDK

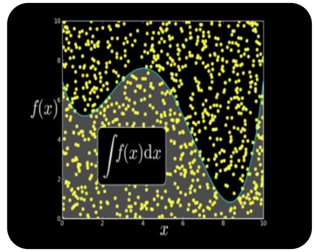
- NVIDIA CUDA Compiler Driver NVCC

- Full information http://docs.nvidia.com/cuda/cuda-compiler-driver-nvcc/#axzz37LQKVSFi

```
nvcc -arch=compute_35 test_CUDA_deviceInfo.cu -o test_CUDA –o deviceInfo
```

Some GPU-accelerated Libraries

NVIDIA cuBLAS



NVIDIA cuRAND

NVIDIA cuSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal Image Processing

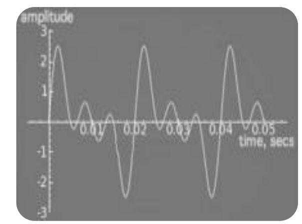
CULA tools

GPU Accelerated Linear Algebra

MAGMA

Matrix Algebra on GPU and Multicore

open source initiative



NVIDIA cuFFT

ROGUE WAVE SOFTWARE

IMSL Library

ArrayFire Matrix Computations

CUSP

Sparse Linear Algebra

Thrust

C++ STL Features for CUDA

Source: https://developer.nvidia.com/cuda-education. (Will Ramey ,NVIDIA Corporation)

Problem HCE: parallelization scheme

- L is a linear differential operator acting on $u(x,y,t)$:

Parallel

↓

- L is a linear differential operator acting on $u(x,y,t)$:

Parallel

Problem HCE: CUDA realization

Initialization: parameters of the problem and the computational scheme are copied in constant memory GPU. Initialization of descriptors: *cuSPARSE* functions

↓

Calculation of array elements lower, upper and main diagonals and right side of SLAEs (1) :

`Kernel_Elements_System_1 <<<blocks, threads>>>()`

↓

Parallel solution of ***(Ny-2)*** SLAEs in the direction ***x*** using

cusparseDgtsvStridedBatch()

↓

Calculation of array elements lower, upper and main diagonals and right side of SLAEs (1) :

`Kernel_Elements_System_2 <<<blocks, threads>>>()`

↓

Parallel solution of ***(Nx-2)*** SLAEs in the direction ***x*** using

cusparseDgtsvStridedBatch()

CUDA realization of parallel algorithm: efficiency of parallelization

Table 1. CUDA realization: Execution time and Acceleration

$N_x \times N_y$	Time CPU [sec]	Time GPU [sec]	Acceleration
500×500	33.976	4.149	8,189
1000×1000	143.693	11.333	12,679
2000×2000	595.058	36.757	16,188
3000×3000	1349.978	100.570	13,423
4000×4000	2406,355	140.718	17,103

- L is a linear differential operator acting on $u(x, y, t)$:

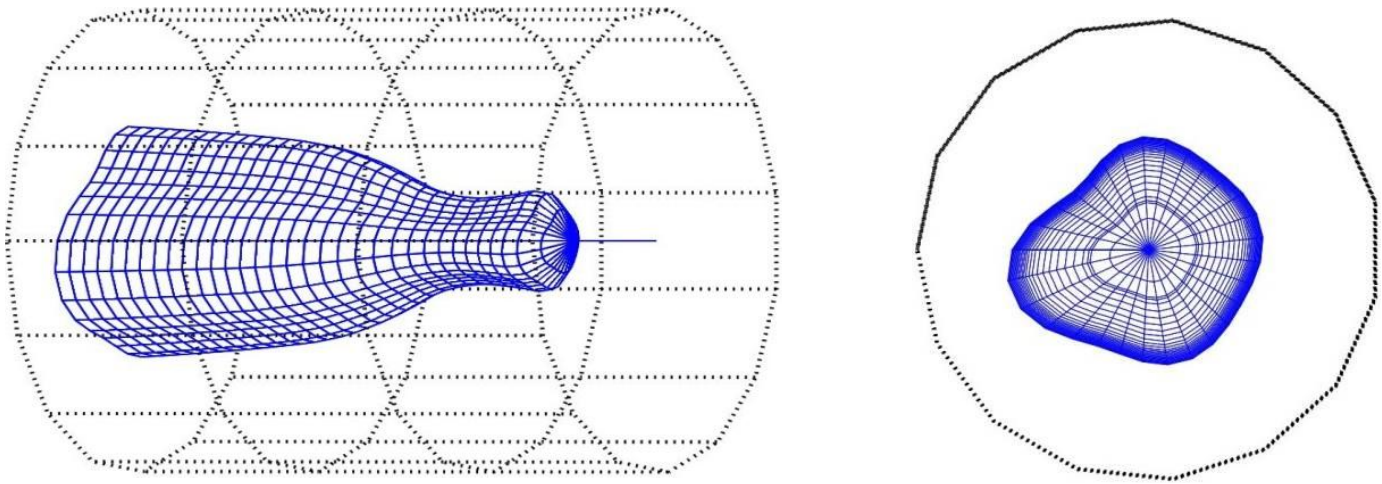
Problem HCE : analysis of results

The best achieved acceleration

- **GPU** ~17 times
- multi-core **CPU (10)** ~7 time
- Intel Xeon Phi Coprocessor ~11 times

Hybrid Programming: MPI+CUDA: on the Example of GIMM FPEIP Complex

GIMM FPEIP : package developed for simulation of thermal processes in materials irradiated by heavy ion beams



Alexandrov E.I., Amirkhanov I.V., Zemlyanaya E.V., Zrelov P.V., Zuev M.I., Ivanov V.V., Podgainy D.V., Sarker N.R., Sarkhadov I.S., Streltsova O.I., Tukhliev Z. K., Sharipov Z.A. (LIT)
Principles of Software Construction for Simulation of Physical Processes on Hybrid Computing Systems (on the Example of GIMM_FPEIP Complex) // Bulletin of Peoples' Friendship University of Russia. Series "Mathematics. Information Sciences. Physics". — 2014. — No 2. — Pp. 197-205.

GIMM FPEIP : package for simulation of thermal processes in materials irradiated by heavy ion beams

To solve a system of coupled equations of heat conductivity which are a basis of the thermal spike model in cylindrical coordinate system

- L is a linear differential operator acting on $u(x, y, t)$:

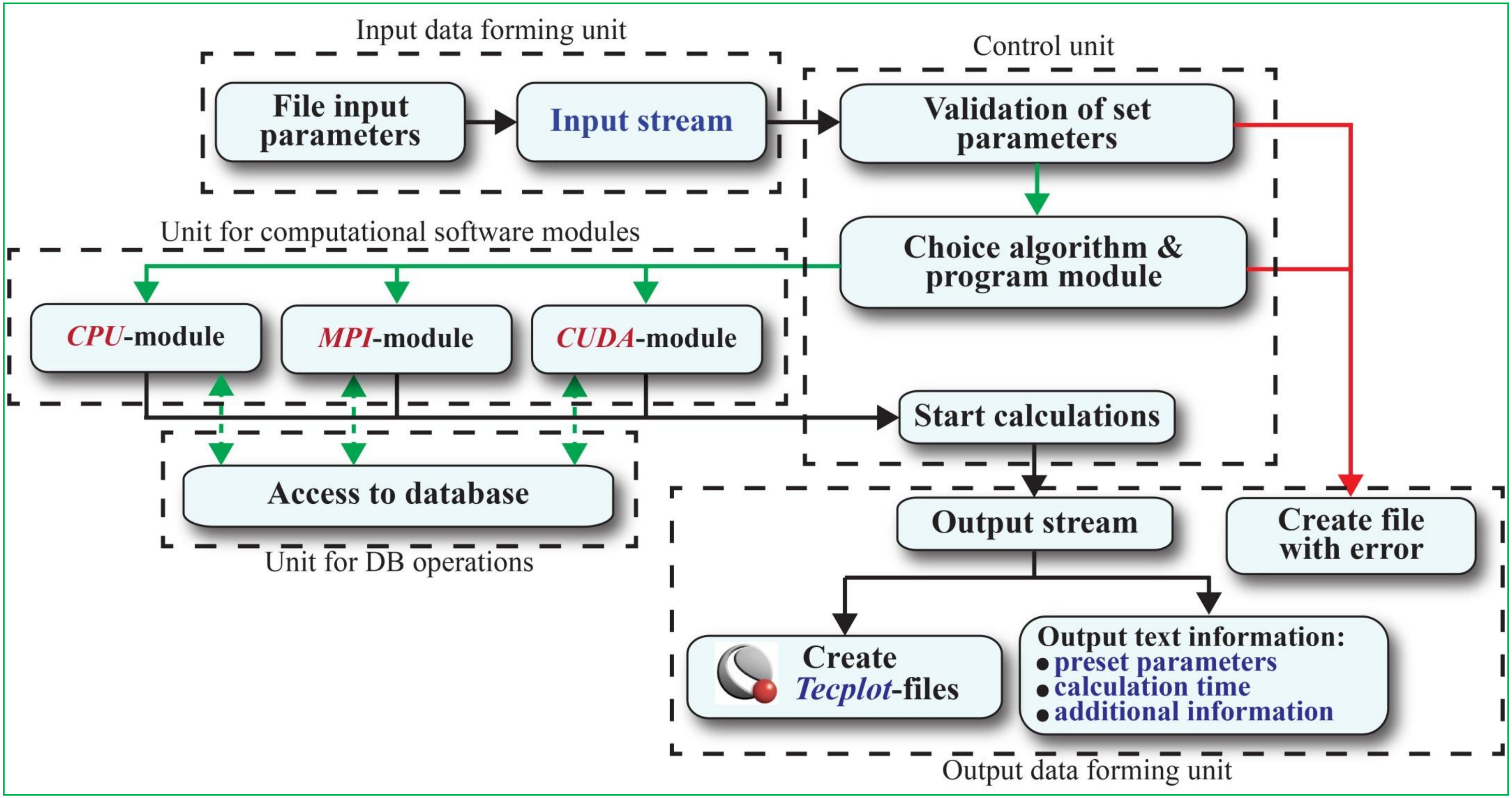
- L is a linear differential operator acting on $u(x, y, t)$:

- L is a linear differential operator acting on $u(x, y, t)$:

Multi-GPU

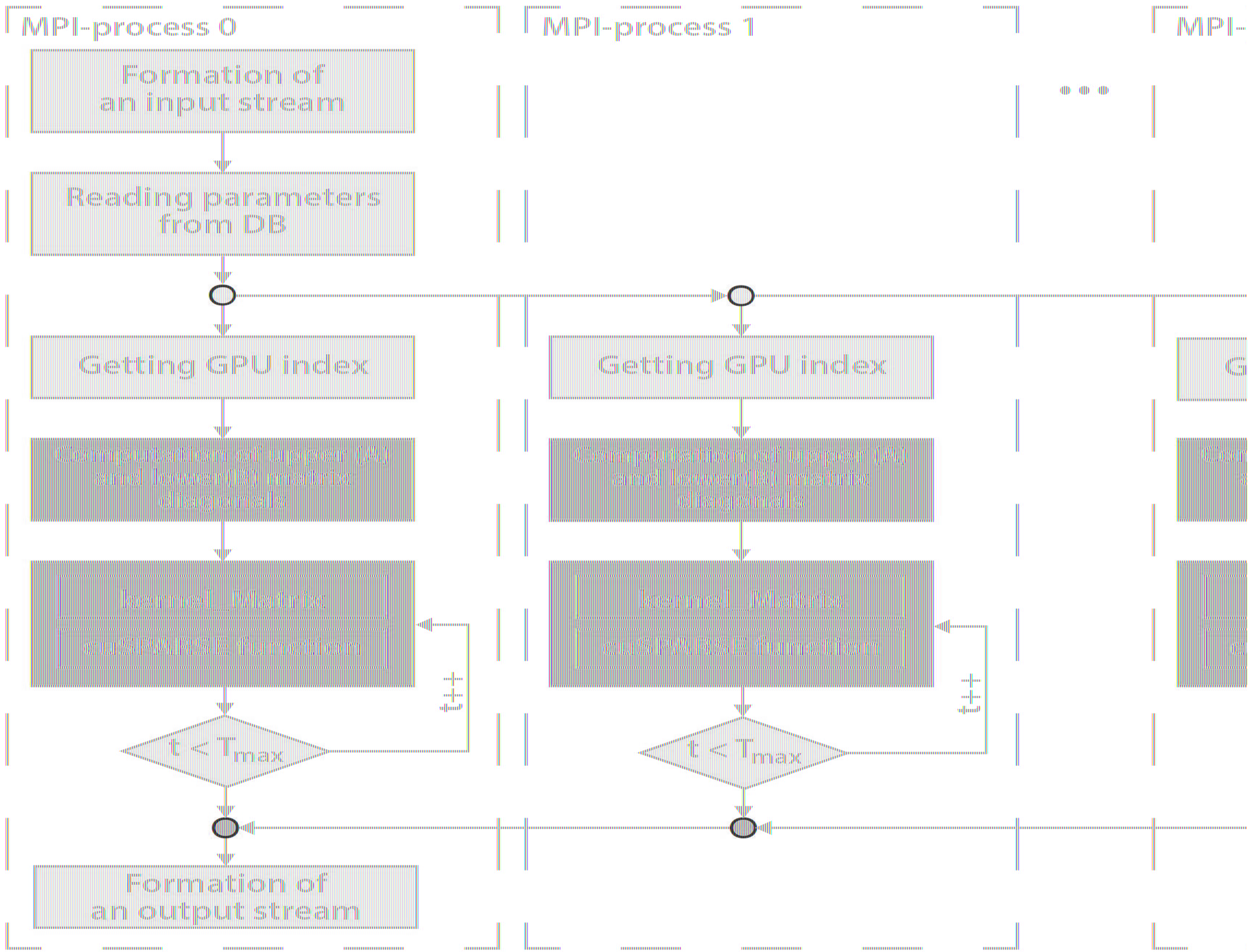
- L is a linear differential operator acting on $u(x, y, t)$:

GIMM FPEIP: Logical scheme of the complex

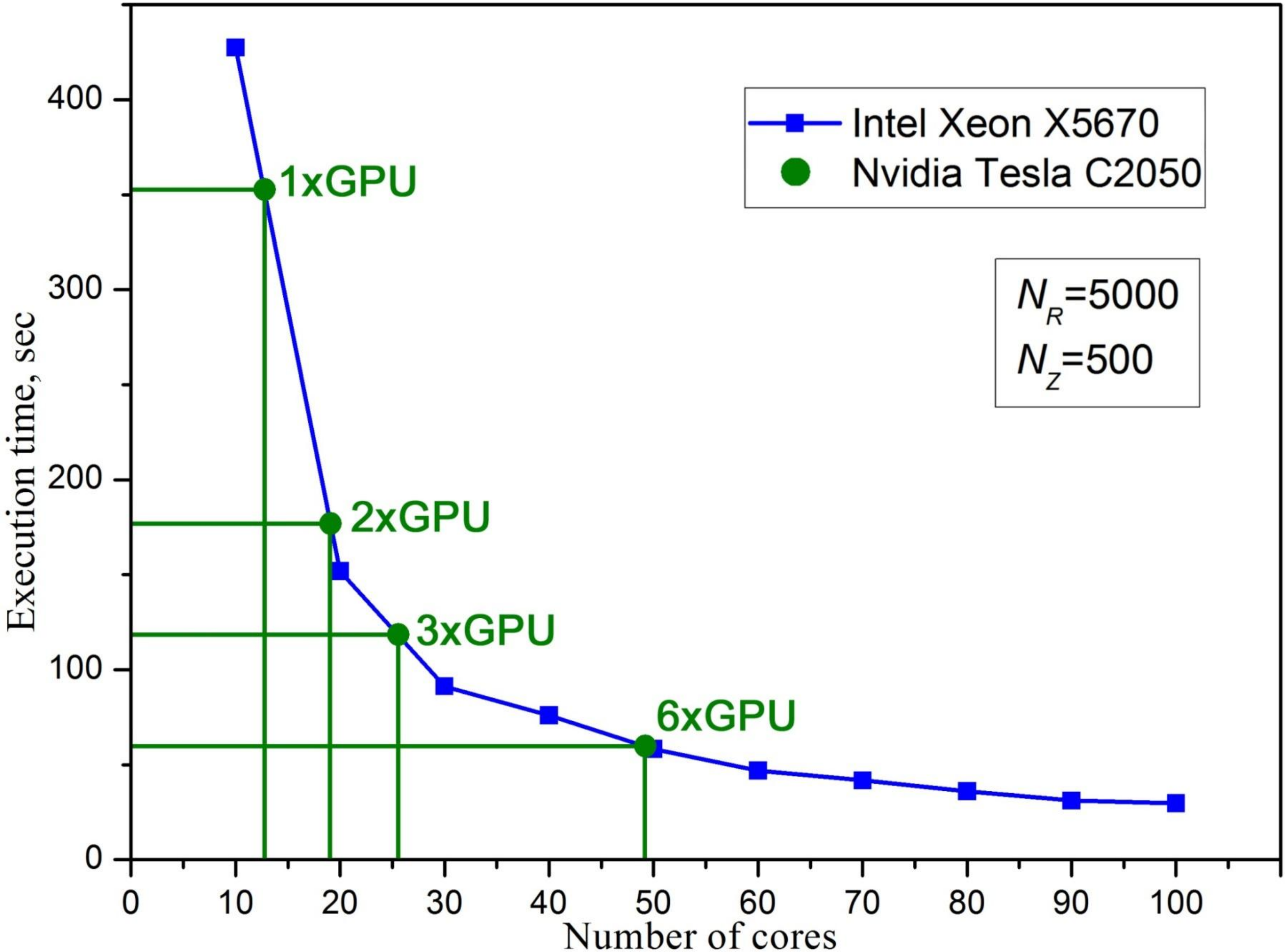


Using Multi-GPUs

L is a linear differential operator acting on $u(x, y, t)$:



MPI, MPI+CUDA (CICC LIT, K100 KIAM)



Hybrid Programming: MPI+OpenMP, MPI+OpenMP+CUDA

MultiConfigurational Ttime Dependnet Hartree (for) Bosons

Ideas, methods, and parallel implementation of the MCTDHB package:

Many-body theory of bosons group in Heidelberg, Germany

http://MCTDHB.org

MCTDHB founders:

Lorenz S. Cederbaum,
Ofir E. Alon,
Alexej I. Streltsov

Since 2013 cooperation with LIT: the development of new hybrid implementations package

The MultiConfigurationalTtimeDependnetHartree (for) Bosons method:
PRL 99, 030402 (2007), PRA 77, 033613 (2008)
It solves TDSE numerically exactly – see for benchmarking PRA 86, 063606 (2012)

Time-Dependent Schrödinger equation governs the physics of trapped ultra-cold atomic clouds

$$i\hbar \frac{\partial}{\partial t} \Psi(\mathrm{x}, t) = \hat{\mathrm{H}}\, \Psi(\mathrm{x}, t)$$

$$\hat{\mathrm{H}} = \sum_{i=1}^{N} \left(-\frac{1}{2m} \nabla_{\vec{r}_i}^2 + V(\vec{r}_i; t) \right) + \sum_{i<j}^{N} \lambda_0 W(\vec{r}_i, \vec{r}_j; t)$$

One has to specify initial condition

$$\Psi(\mathrm{x}, t=0) = \Psi(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_N, t=0)$$

and propagate $\Psi(\mathbf{x}, t) \longrightarrow \Psi(\mathbf{x}, t + \Delta t)$

To solve the Time-Dependent Many-Boson Schrödinger Equation

we apply the MultiConfigurational TtimeDependnetHartree (for) Bosons method:

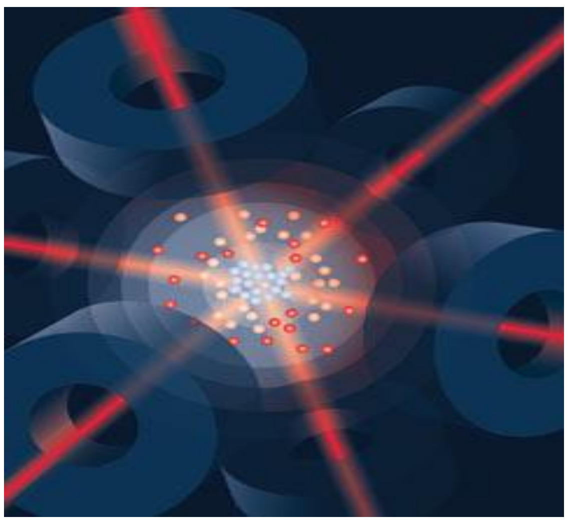
PRL 99, 030402 (2007), PRA 77, 033613 (2008)

It solves TDSE numerically exactly – see for benchmarking PRA 86, 063606 (2012)

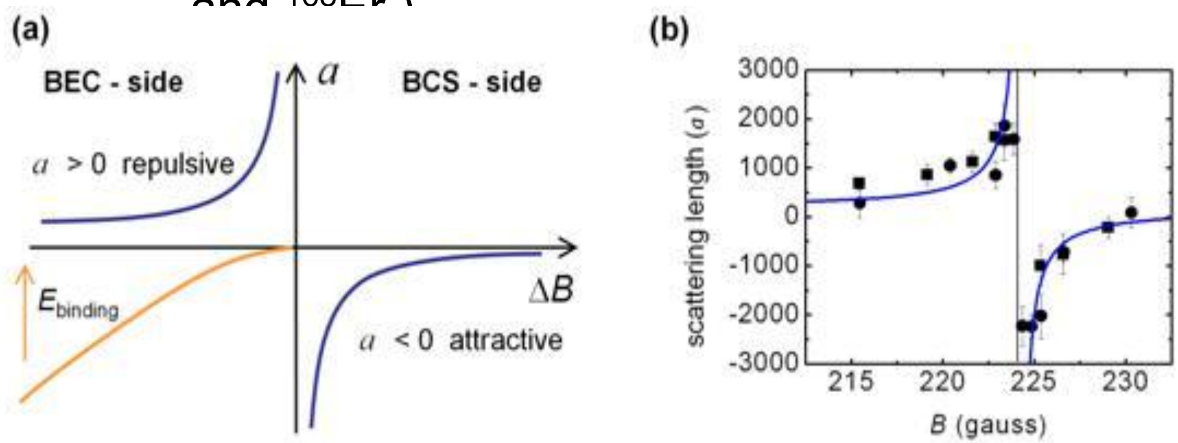
All the terms of the Hamiltonian are under experimental control and can be manipulated

$$\hat{H} = \sum_{i=1}^{N}\left(-\frac{1}{2m}\nabla^2_{r_i} + V(r_i; t)\right) + \sum_{i<j}^{N}\lambda_0 W(r_i, r_j; t)$$

BECs of alkaline, alkaline earth, and lanthanoid atoms
(^{7}Li, ^{23}Na, ^{39}K, ^{41}K, ^{85}Rb, ^{87}Rb, ^{133}Cs, ^{52}Cr, ^{40}Ca, ^{84}Sr, ^{86}Sr, ^{88}Sr, ^{174}Yb, ^{164}Dy, and ^{168}Er.)



Magneto-optical trap $\rightarrow \mathbf{V}(r,t)$



The interatomic interaction can be widely varied with a magnetic Feshbach resonance… (Greiner Lab at Harvard.)

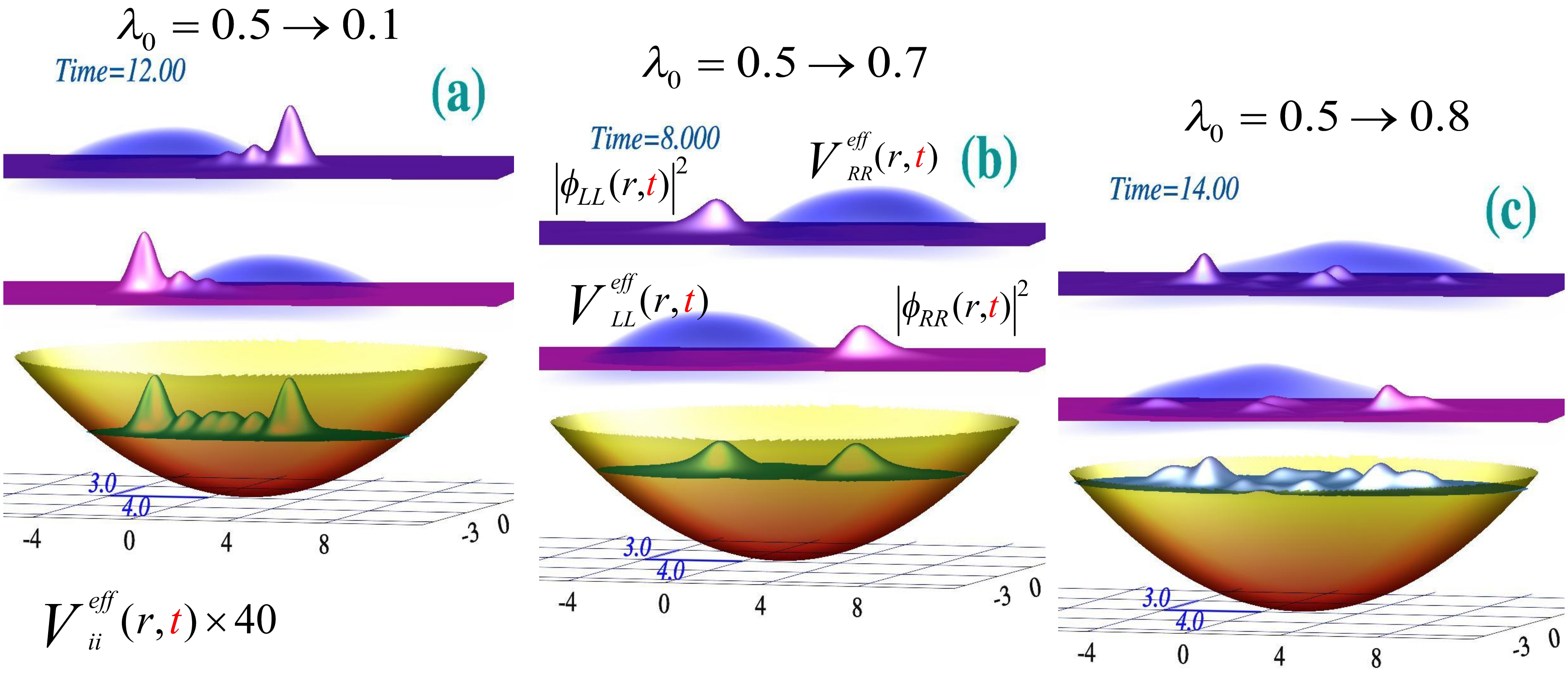
1D-2D-3D: Control on dimensionality by changing the aspect ratio of the

$$\mathbf{V}(x,y,z) = \tfrac{1}{2}m\omega_x^2x^2 + \tfrac{1}{2}m\omega_y^2y^2 + \tfrac{1}{2}m\omega_z^2z^2$$

Dynamics N=100: sudden displacement of trap and sudden quenches of the repulsion in 2D

arXiv:1312.6174

$$\mathbf{V}(x,y)=\tfrac{1}{2}x^2+\tfrac{3}{2}y^2\rightarrow\mathbf{V}(x-1.5,y-0.5)$$



Two generic regimes: (i) non-violent (under-a-barrier) and (ii) Explosive (over-a-barrier)

Conclusion

- Modern development of computer technologies (multi-core processors, GPU , coprocessors and other) require the development of new approaches and technologies for parallel programming.
- Effective use of high performance computing systems allow accelerating of researches, engineering development and creation of a specific device.

Thank you for attention!