Programming Infrastructure of Heterogeneous Computing Based on OpenCL and its Applications

Anton Efremov, antonyef@mail.ru
Pavel Bogdanov, bogdanov@niisi.msk.ru

Heterogeneous parallel programming group
Institute of System Research
Russian Academy of Science
http://www.niisi.ru
What it's about?

- Hybrid System Architecture (HSA)
- Heterogeneous programming
Introduction: what and why?

- The methodology of hybrid cluster programming
- Set of distributed tasks for verification with world leaders and cross-optimization
## Top 500 — November 2012

<table>
<thead>
<tr>
<th>Rank</th>
<th>Site</th>
<th>Computer</th>
<th>Total Cores</th>
<th>Accelerator/ Co-Processor Cores</th>
<th>Rmax (TFlop/s)</th>
<th>Rpeak (TFlop/s)</th>
<th>Eff(%)</th>
<th>Power (kW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DOE/SC/Oak Ridge National Laboratory United States</td>
<td><em>Titan</em> — Cray XK7, Opteron 6274 16C 2.20GHz, Cray Gemini interconnect, NVIDIA K20x, Cray Inc.</td>
<td>560640</td>
<td>261632</td>
<td>17590000</td>
<td>27112550</td>
<td>64,88</td>
<td>8209</td>
</tr>
<tr>
<td>2</td>
<td>DOE/NNSA/LLNL United States</td>
<td><em>Sequoia</em> — BlueGene/Q, Power BQC 16C 1.60 GHz, Custom, IBM</td>
<td>1572864</td>
<td>0</td>
<td>16324751</td>
<td>20132659,2</td>
<td>81,09</td>
<td>7890</td>
</tr>
<tr>
<td>3</td>
<td>RIKEN Advanced Institute for Computational Science (AICS) Japan</td>
<td><em>K computer</em>, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu</td>
<td>705024</td>
<td>0</td>
<td>10510000</td>
<td>11280384</td>
<td>93,17</td>
<td>12659,89</td>
</tr>
<tr>
<td>4</td>
<td>DOE/SC/Argonne National Laboratory United States</td>
<td><em>Mira</em> — BlueGene/Q, Power BQC 16C 1.60GHz, Custom, IBM</td>
<td>786432</td>
<td>0</td>
<td>8162376</td>
<td>10066330</td>
<td>81,09</td>
<td>3945</td>
</tr>
<tr>
<td>5</td>
<td>Forschungszentrum Juelich (FZJ) Germany</td>
<td><em>JUQUEEN</em> — BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect, IBM</td>
<td>393216</td>
<td>0</td>
<td>4141180</td>
<td>5033165</td>
<td>82,28</td>
<td>1970</td>
</tr>
<tr>
<td>6</td>
<td>Leibniz Rechenzentrum Germany</td>
<td><em>SuperMUC</em> — iDataPlex DX360M4, Xeon E5-2680 8C 2.7GHz, Infiniband FDR, SMUC</td>
<td>147456</td>
<td>0</td>
<td>2897000</td>
<td>3185050</td>
<td>90,96</td>
<td>3422,67</td>
</tr>
<tr>
<td>7</td>
<td>Texas Advanced Computing Center/Univ. of Texas United States</td>
<td><em>Stampede</em> — PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi, Dell</td>
<td>204900</td>
<td>112500</td>
<td>2660290</td>
<td>3958965</td>
<td>67,2</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>National Supercomputing Center in Tianjin China</td>
<td><em>Tianhe-1A</em> — NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050, NUDT</td>
<td>186368</td>
<td>100352</td>
<td>2566000</td>
<td>4701000</td>
<td>54,58</td>
<td>4040</td>
</tr>
<tr>
<td>9</td>
<td>CINECA Italy</td>
<td><em>Fermi</em> — BlueGene/Q, Power BQC 16C 1.60GHz, CI, IBM</td>
<td>163840</td>
<td>0</td>
<td>1725492</td>
<td>2097152</td>
<td>82,28</td>
<td>821,88</td>
</tr>
<tr>
<td>10</td>
<td>IBM Development Engineering United States</td>
<td><em>DARPA Trial Subset</em> — Power 775, POWER7 8C 3.836GHz, Custom Interconnect, IBM</td>
<td>63360</td>
<td>0</td>
<td>1515000</td>
<td>1944391,68</td>
<td>77,92</td>
<td>3575,63</td>
</tr>
</tbody>
</table>
**Hybrid computers**

- **2006**: IBM Cell BE
- **2007**: GPGPU
- **2010**: FPGA
- **2010-2012**: the era of post-PC tablet revolution
- **2012**: Intel MIC
- **2013-...**: the era of micro servers
## Summary table of popular architectures

<table>
<thead>
<tr>
<th>Processor</th>
<th>Type</th>
<th>GFLOPS (32bit)</th>
<th>GFLOPS (64bit)</th>
<th>Watt (TDP)</th>
<th>GFLOPS/Watt (32bit)</th>
<th>GFLOPS/Watt (64bit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adapteva Epiphany-IV</td>
<td>Epiphany</td>
<td>100</td>
<td>N/A</td>
<td>2</td>
<td>50</td>
<td>N/A</td>
</tr>
<tr>
<td>Movidius Myriad</td>
<td>ARM SoC: LEON3+SHAVE</td>
<td>15.28</td>
<td>N/A</td>
<td>0.32</td>
<td>48</td>
<td>N/A</td>
</tr>
<tr>
<td>ZiiLabs</td>
<td>ARM SoC</td>
<td>58</td>
<td>N/A</td>
<td>3</td>
<td>20</td>
<td>N/A</td>
</tr>
<tr>
<td>Nvidia Tesla K10</td>
<td>X86 GPU</td>
<td>4577</td>
<td>190</td>
<td>225</td>
<td>20.34</td>
<td>0.84</td>
</tr>
<tr>
<td>ARM + MALI T604</td>
<td>ARM SoC</td>
<td>8 + 68</td>
<td>N/A</td>
<td>4</td>
<td>19</td>
<td>N/A</td>
</tr>
<tr>
<td>NVidia GTX 690</td>
<td>X86 GPU x 2</td>
<td>5621</td>
<td>234?</td>
<td>300</td>
<td>18.74</td>
<td>0.78</td>
</tr>
<tr>
<td>GeForce GTX 680</td>
<td>X86 GPU</td>
<td>3090</td>
<td>128</td>
<td>195</td>
<td>15.85</td>
<td>0.65</td>
</tr>
<tr>
<td>AMD Radeon HD 7970 GHz</td>
<td>X86 GPU</td>
<td>4300</td>
<td>1075</td>
<td>300+</td>
<td>14.3</td>
<td>3.58</td>
</tr>
<tr>
<td>Nvidia Tesla K20X (Kepler)</td>
<td>X86 GPU</td>
<td>3950</td>
<td>1310</td>
<td>235</td>
<td>16,8</td>
<td>5,5</td>
</tr>
<tr>
<td>Intel Knight's Corner (Xeon Phi)</td>
<td>X87?</td>
<td>2000</td>
<td>1000</td>
<td>200</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>AMD A10-5800K + HD 7660D</td>
<td>X86 SoC</td>
<td>121 + 614</td>
<td>60</td>
<td>100</td>
<td>7.35</td>
<td>0.6</td>
</tr>
<tr>
<td>Intel Core i7-3770 + HD4000</td>
<td>X86 SoC</td>
<td>225 + 294.4</td>
<td>112 + 73.6</td>
<td>77</td>
<td>6.74</td>
<td>2.41</td>
</tr>
<tr>
<td>NVIDIA CARMA (complete board)</td>
<td>ARM + GPU</td>
<td>70 + 200</td>
<td>35</td>
<td>40</td>
<td>6.75</td>
<td>0.87</td>
</tr>
<tr>
<td>IBM Power A2</td>
<td>Power CPU</td>
<td>408</td>
<td>204</td>
<td>55</td>
<td>7.44</td>
<td>3.72</td>
</tr>
</tbody>
</table>
Scheme of 10TFlops DP node

- 4 CPU, 64 cores
- 256 Gb RAM
- PCIe expansion 8 GPU
- SAS RAID → 50 Tb
- Interconnect

Per node:
- TFlops ~ 10 = 8*1260+4*110 (double precision)
- Power ~ 2.7 kW
- Price ~ 15000 $
The main problem of software development for modern supercomputers
Who will win: SMP CPU or HYBRID?

Target architecture? How to develop the software?
What does the HPC industry for software developers?

Low-level models:

Classic scheme: MPI/OpenMP

Hybrid node: MPI/OpenMP

CUDA vs OpenCL

- Proprietary model
- Works only with nVidia GPUs

- Open standard
- Supports a wide class of devices: CPU, GPU, MIC, FPGA, CELL and more
- Supported by Khronos group. AMD, NVidia, Intel, IBM and else provide OpenCL drivers
What does the HPC industry for software developers?

Hi-level models:

**Opa pragma style:**

- **OpenACC** - programming standard for parallel computing
  - Cray, CAPS, Nvidia, PGI

- **OpenHMPP** - Hybrid Multicore Parallel Programming
  - INRIA, CNRS,
  - University of Rennes, INSA of Rennes

**Task scheduling:**

- **StarPU** - task programming library for hybrid architectures
  - INRIA, France

- **CGCM** - CPU-GPU Communication Manager
  - Princeton University, Princeton, NJ, USA

...
Many years of MPI/OpenMP, and what is further?

Classic approach

MPI/OpenMP code

Library API

CPU numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

CPU 1

CPU N
Programming models

Basic hybrid approach

- **New(!)** MPI/OpenMP code
- **New(!)** Library API
- **New(!)** DEVICE numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

Classic approach

- MPI/OpenMP code
- **Library API**
- **CPU** numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

Heterogeneous parallel programming group ISR RAS
http://www.niisi.ru
Programming models

Basic hybrid approach

New(!) MPI/OpenMP code

New(!) Library API

New(!) DEVICE numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

CPU 1       GPU 1       MIC 1

...       ...       ...

CPU N       GPU M       MIC L

Classic approach

MPI/OpenMP code

Library API

CPU numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

CPU 1       CPU 1

...       ...

CPU N       CPU N

Probably better way

Minimize changes MPI/OpenMP code

Standard Library API

Scheduler-adaptor

New(!) DEVICE numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

CPU 1       GPU 1       MIC 1

...       ...       ...

CPU N       GPU M       MIC L

New(!) Library API

New(!) DEVICE numerical libraries: BLAS, LAPACK, SparseBLAS, FFT, ILU, AMG,…

CPU 1       GPU 1       MIC 1

...       ...       ...

CPU N       GPU M       MIC L
Programming infrastructure of heterogeneous computing
Scheduler: main ideas

- Provide unified infrastructure to develop programs for hybrid clusters

- Priorities:
  - Simple logical model
  - Scalability
  - Minimal code changes
  - Automatic transfer overlap when possible
Scheduler: concept scheme

Host program (CPU thread 1)

Scheduler

LD EXEC ST

Device 1

r1 r2 → ALU → r4
r3
r5

Host program (CPU thread i)

Scheduler

LD EXEC ST

Device N

r1 r2 → ALU → r4
r3
r5

Host program (CPU thread M)

Scheduler

LD EXEC ST

Host program

(CPU thread i)

Scheduler

LD EXEC ST

Device N

r1 r2 → ALU → r4
r3
r5

Host program

(CPU thread M)

Scheduler

LD EXEC ST

Host program

(CPU thread 1)
Scheduler: concepts

- Register – a region in device global memory
- Instruction – compute kernel for the device
- Logically, a device is represented as a coprocessor running instructions on registers
Scheduler: memory commands

- **LD**: load a n-dimensional region from host memory to a register on device

- **ST**: store a region from device host
• EXEC command launches an arbitrary kernel on a device

• Arguments – registers and scalar values
Scheduler: program

- A program for a scheduler is a graph of dependencies between commands.

- Command is ready when all commands it depends on are ready.
• Scheduler has 3 queues for LD, ST and EXEC commands

• Independent commands from different queues run simultaneously

• EXEC runs in-order, LD and ST – asynchronously
Scheduler: writing a program

1. Write compute kernels

2. Arrange dependencies and fill queues

3. Start execution
Model tasks
### Equipment

**Reference CPU:**

- **Intel Xeon X5670 2.93GHz**
  - 12 Gflops, 32 Gb/s

**GPU devices:**

- **nVidia Tesla C2050**
  - 515 Gflops, 144 Gb/s

- **nVidia GeForce GTX480**
  - 164 Gflops, 177 Gb/s

- **AMD Radeon 5870**
  - 544 Gflops, 153 Gb/s

- **AMD Radeon 6970**
  - 683 Gflops, 176 Gb/s

- **AMD Radeon 7970**
  - 925 Gflops, 264 Gb/s

- **Supercomputer K100 (www.kiam.ru)**
Our testing platform: 8.3 TFlop/s DP, 2.8 kW

Performance (double-precision, DP):
2xXeon E-2670 = 0.330 TFlops
8x[AMD Radeon 7970] = 8.08 TFlops
Total: ~8.4 TFlops DP (~33 TFlops SP)
PowerC: ~ 2.8kW
Our testing platform

PCI Express Expansion Systems – Cyclone
High Performance Linpack

- Resource-intensive procedures of dense linear algebra:
  - **DGEMM** — general matrix multiplication
  - **DTRSM** — solution of the triangular systems of linear equations
  - **DGETRF** — LU-decomposition
DGEMM: algorithm

for (i, j, l)
  EXEC padding r0->ima
  EXEC padding r1->imb
  if (l == 0)
    EXEC padding r2->imc
    EXEC unpadding imc->r4
  endif
endif
EXEC dgemm ima, imb, imc
if (l == 0)
  LD Cij->r0
endif
ST r4->Cij
for (size_t iter = 0; iter < nsteps; ++iter) {

    /* calculate offsets & block parameters for all matrices */
    ldr0   = new LdCmd(r0, srcA, lda, rowsA, colsA);
    ldr1   = new LdCmd(r1, srcB, ldb, rowsB, colsB);
    padr0a = new PadCmd(pad_kernel_a, r0, ima, rowsA, colsA, imah, imaw);
    padr1b = new PadCmd(pad_kernel_b, r1, imb, rowsB, colsB, imbh, imbw);
    if (l == 0) {
        ldr2   = new LdCmd(r2, srcC, ldc, rowC, colC);
        padr2c = new PadCmd(pad_kernel_c, r2, imc, rowC, colC, imch, imcw);
    }
    gemm   = new GemmCmd(gemm_kernel, imch, imcw, imaw, alpha, ima, imb, beta, imc);
    if (l == block_K - 1) {
        unpadr5 = new PadCmd(unpad_kernel_c, imc, r2, rowC, colC, imch, imcw);
        str5   = new StCmd(r2, srcC, ldc, rowC, colC);
    }
    /* add dependencies */
    /* add tasks to scheduler */
}
DGEMM scalability

BLAS DGEMM (NxNxN)

<table>
<thead>
<tr>
<th>nDev</th>
<th>Rpeak</th>
<th>Rmax</th>
<th>Scl</th>
<th>Eff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1010</td>
<td>702</td>
<td>1</td>
<td>69%</td>
</tr>
<tr>
<td>2</td>
<td>2020</td>
<td>1380</td>
<td>1.97</td>
<td>68%</td>
</tr>
<tr>
<td>3</td>
<td>3030</td>
<td>2020</td>
<td>2.88</td>
<td>66%</td>
</tr>
<tr>
<td>4</td>
<td>4040</td>
<td>2700</td>
<td>3.84</td>
<td>67%</td>
</tr>
<tr>
<td>5</td>
<td>5050</td>
<td>3270</td>
<td>4.66</td>
<td>65%</td>
</tr>
<tr>
<td>6</td>
<td>6060</td>
<td>4000</td>
<td>5.79</td>
<td>66%</td>
</tr>
<tr>
<td>7</td>
<td>7070</td>
<td>4600</td>
<td>6.55</td>
<td>65%</td>
</tr>
<tr>
<td>8</td>
<td>8080</td>
<td>5140</td>
<td>7.32</td>
<td>64%</td>
</tr>
</tbody>
</table>

2 x [Xeon E5-2670 8 cores 2.6Ghz ]
+ 8 x [AMD Radeon 7970 1010Mhz]
= ~8.4 Tflops Double Precision

~65% efficiency on 8 GPU
DGETRF scalability

BLAS DGETRF (NxN)

<table>
<thead>
<tr>
<th>nDev</th>
<th>Rpeak</th>
<th>Rmax</th>
<th>Scl</th>
<th>Eff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1340</td>
<td>656</td>
<td>1</td>
<td>49%</td>
</tr>
<tr>
<td>2</td>
<td>2350</td>
<td>1223</td>
<td>1.8</td>
<td>52%</td>
</tr>
<tr>
<td>3</td>
<td>3360</td>
<td>1638</td>
<td>2.5</td>
<td>49%</td>
</tr>
<tr>
<td>4</td>
<td>4370</td>
<td>2030</td>
<td>3.1</td>
<td>46%</td>
</tr>
<tr>
<td>5</td>
<td>5380</td>
<td>2250</td>
<td>3.4</td>
<td>42%</td>
</tr>
<tr>
<td>6</td>
<td>6390</td>
<td>2400</td>
<td>3.7</td>
<td>38%</td>
</tr>
</tbody>
</table>

2 x [Xeon E5-2670 8 cores 2.6Ghz ]
+ 8 x [AMD Radeon 7970 1010Mhz]
= ~8,4 Tflops Double Precision
 ISR RAS vs MAGMA

Collaborating Partners:
University of California, Berkeley
University of Colorado, Denver
INRIA Bordeaux - Sud Ouest
INRIA Saclay/University Paris-Sud
KAUST

MAGMA
Matrix Algebra on GPU and Multicore Architectures

- MAGMA for CUDA
- clMAGMA for OpenCL
- MAGMA MIC for Intel Xeon Phi

<table>
<thead>
<tr>
<th></th>
<th>PEAK 1 DEV Tflop/s DP</th>
<th>DGERMM Tflop/s DP</th>
<th>DGERMM Eff %</th>
<th>DGETRF Eff %</th>
<th>SCL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAGMA-MIC (Xeon Phi)</td>
<td>1.04</td>
<td>0.83</td>
<td>84</td>
<td>60</td>
<td>4</td>
</tr>
<tr>
<td>MAGMA-CUDA (Kepler K20X)</td>
<td>1.32</td>
<td>1.2</td>
<td>92</td>
<td>73</td>
<td>3</td>
</tr>
<tr>
<td>ISR RAS (Radeon HD 7970 Ghz Ed)</td>
<td>1.01</td>
<td>0.7</td>
<td>70</td>
<td>65</td>
<td>8</td>
</tr>
</tbody>
</table>
**ISR RAS vs MAGMA**

**Device peak performance**

- Radeon HD 7970 Ghz Ed
- Xeon Phi
- Kepler K20X

**Device DGEMM performance**

- ISR RAS (Radeon HD 7970 Ghz Ed)
- MAGMA-MIC (Xeon Phi)
- MAGMA-CUDA (Kepler K20X)

**ISR RAS LU in DP on AMD Radeon 7970 Ghz Ed**
- up to 6 devices
  - Matrix size
  - GFlops
  - 2400
  - 2250
  - 2030
  - 1638
  - 1223
  - 656

**MAGMA LU in DP on Intel MIC**
- up to 4 devices
  - Matrix size
  - GFlops
  - 2150
  - 1750
  - 1200
  - 600

**MAGMA LU in DP on Kepler 20X**
- up to 2 devices
  - Matrix size
  - GFlops
  - 1750
  - 950

Host: Sandy Bridge (2x8 @ 2.6 Ghz) DP Peak 332 GFlop/s
Sphere in a supersonic flow

Programming infrastructure application: model CFD task
CFD is highly resource-consuming

- High-ordered schemes for accuracy
- The large size of computational domain and high scale differential
- High space and time resolution
- Long period for time integration
Simplified algorithm

Model task: sphere in supersonic flow (M = 2.75)

We consider a compressible inviscid flow:

- Euler equations
- Finite volume approach
- Unstructured hybrid mesh (tetrahedrons, hexahedrons, quadrangular pyramids and triangular prisms)
- Roe scheme
- The explicit Runge-Kutta scheme up to 4th order of accuracy
Performance on one device

- **Time spent on model task (0.4 Millions of tetrahedrons)**
- **Average on 100 time steps**
- **We measured the following operations:**
  - Convective flux through faces of control volumes (Roe scheme)
  - Calculating the sum of fluxes through control volumes (GPU only)
  - Border conditions
  - Integration steps (Runge-Kutta)
- **Compare 1 CPU (1 core) with 1 GPU (all cores)**
- **Floating operations in double precision**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Intel Xeon E5504 2.0GHz</th>
<th>AMD Opteron 6176 2.3GHz</th>
<th>Intel Xeon X5670 2.93GHz</th>
<th>Intel Xeon E5-2670 2.6GHz</th>
<th>NVidia C1060</th>
<th>NVidia GTX470</th>
<th>NVidia C2050</th>
<th>ATI Radeon 5870</th>
<th>AMD Radeon 6970</th>
<th>AMD Radeon 7970 Ghz Edition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Roe scheme</td>
<td>0.255</td>
<td>0.222</td>
<td>0.172</td>
<td>0.117</td>
<td>0.0095</td>
<td>0.0053</td>
<td>0.0056</td>
<td>0.0041</td>
<td>0.0031</td>
<td>0.0012</td>
</tr>
<tr>
<td>Flux sum</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>0.0043</td>
<td>0.0033</td>
<td>0.0026</td>
<td>0.0030</td>
<td>0.0022</td>
<td>0.0011</td>
</tr>
<tr>
<td>Border conditions</td>
<td>0.0011</td>
<td>0.0009</td>
<td>0.0008</td>
<td>0.0055</td>
<td>0.00045</td>
<td>0.00021</td>
<td>0.0003</td>
<td>0.0004</td>
<td>0.00028</td>
<td>0.00007</td>
</tr>
<tr>
<td>Runge-Kutta</td>
<td>0.029</td>
<td>0.025</td>
<td>0.022</td>
<td>0.020</td>
<td>0.0015</td>
<td>0.00103</td>
<td>0.0014</td>
<td>0.0016</td>
<td>0.0019</td>
<td>0.00083</td>
</tr>
<tr>
<td>Total</td>
<td>0.285</td>
<td>0.247</td>
<td>0.195</td>
<td>0.138</td>
<td>0.0157</td>
<td>0.00984</td>
<td>0.0099</td>
<td>0.0091</td>
<td>0.0075</td>
<td>0.0033</td>
</tr>
</tbody>
</table>
Scheduler programm for one device

for \( i = 0; i < \text{niter}; i++ \)
Scheduler programm for one device
1 core Xeon E5-2670 is 1.79x faster than 1 core Opteron 6176 SE

16 cores Xeon E5-2670 OpenCL is 2.42x faster than 24 cores Opteron 6176 SE OpenCL

1 AMD Radeon 7970 is 75x faster than 1 core Opteron 6176 SE

1 AMD Radeon 7970 is 42x faster than 1 core Xeon E5-2670

1 AMD Radeon 7970 is 3x faster than nVidia M2050
### 8 GPU Radeon 7970 vs 16 CPU Cores E5-2670 via OpenCL Speed Up

#### Num of devices

<table>
<thead>
<tr>
<th>e^G</th>
<th>2 GPU</th>
<th>3 GPU</th>
<th>4 GPU</th>
<th>5 GPU</th>
<th>6 GPU</th>
<th>7 GPU</th>
<th>8 GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5M</td>
<td>1.84</td>
<td>2.30</td>
<td>5.00</td>
<td>4.38</td>
<td>4.00</td>
<td>3.68</td>
<td>3.26</td>
</tr>
<tr>
<td>1M</td>
<td>3.05</td>
<td>2.79</td>
<td>6.84</td>
<td>7.79</td>
<td>8.17</td>
<td>6.32</td>
<td>5.98</td>
</tr>
<tr>
<td>2M</td>
<td>4.12</td>
<td>4.67</td>
<td>5.83</td>
<td>11.67</td>
<td>12.73</td>
<td>11.29</td>
<td>10.61</td>
</tr>
<tr>
<td>4M</td>
<td>5.20</td>
<td>7.22</td>
<td>5.00</td>
<td>13.27</td>
<td>15.29</td>
<td>15.29</td>
<td>14.61</td>
</tr>
<tr>
<td>8M</td>
<td>7.69</td>
<td>9.11</td>
<td>11.71</td>
<td>15.38</td>
<td>18.92</td>
<td>20.50</td>
<td>17.57</td>
</tr>
<tr>
<td>16M</td>
<td>*</td>
<td>12.16</td>
<td>11.15</td>
<td>19.11</td>
<td>16.21</td>
<td>12.44</td>
<td>16.21</td>
</tr>
<tr>
<td>32M</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>15.18</td>
<td>14.23</td>
<td>18.22</td>
</tr>
<tr>
<td>40M</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>20.69</td>
</tr>
</tbody>
</table>
The efficiency of parallelization (1=theoretical estimation)

<table>
<thead>
<tr>
<th>Grid size</th>
<th>2 GPU</th>
<th>3 GPU</th>
<th>4 GPU</th>
<th>5 GPU</th>
<th>6 GPU</th>
<th>7 GPU</th>
<th>8 GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5 M</td>
<td>0.20</td>
<td>0.16</td>
<td>0.29</td>
<td>0.21</td>
<td>0.15</td>
<td>0.12</td>
<td>0.11</td>
</tr>
<tr>
<td>1 M</td>
<td>0.30</td>
<td>0.18</td>
<td>0.33</td>
<td>0.30</td>
<td>0.24</td>
<td>0.18</td>
<td>0.15</td>
</tr>
<tr>
<td>2 M</td>
<td>0.38</td>
<td>0.29</td>
<td>0.28</td>
<td>0.45</td>
<td>0.40</td>
<td>0.29</td>
<td>0.24</td>
</tr>
<tr>
<td>4 M</td>
<td>0.52</td>
<td>0.50</td>
<td>0.27</td>
<td>0.55</td>
<td>0.53</td>
<td>0.46</td>
<td>0.37</td>
</tr>
<tr>
<td>8 M</td>
<td>0.72</td>
<td>0.56</td>
<td>0.05</td>
<td>0.06</td>
<td>0.60</td>
<td>0.58</td>
<td>0.43</td>
</tr>
<tr>
<td>16 M</td>
<td>*</td>
<td>0.73</td>
<td>0.50</td>
<td>0.68</td>
<td>0.48</td>
<td>0.30</td>
<td>0.33</td>
</tr>
<tr>
<td>32 M</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0.57</td>
<td>0.50</td>
<td>0.48</td>
</tr>
<tr>
<td>40 M</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>0.60</td>
<td>0.40</td>
</tr>
</tbody>
</table>

- 0< ... < 20%
- 20% < ... < 30%
- 30% < ... < 50%
- 50% < ... < 100%
The efficiency of the parallelization on a supercomputer K100

Speedups on K100 supercomputer for a mesh with 4 millions of cells (left) starting from one GPU and for a mesh with 16 millions of cells (right) starting from one computing node.
SpMV:
Sparse Matrix-Vector multiplication
• Sliced ELLPACK format: matrix cut into “slices”, each slice stored in ELL format

• Matrix stored in:
  - Slice start
  - Column
  - Value

Anton Lokhmotov, ARM
Implementing Sparse Matrix Vector Product in OpenCL
OpenCL tutorial, HiPEAC’11
SpMV: Test matrices

- Dense
- Protein
- FEM/Sphere
- FEM/Cantilever
- Wind tunnel
- FEM/Ship
- Economics
- Epidemiology
- FEM/Accelerator
- Circuit
- Webbase
- LP

- Dense matrix in sparse format
- Protein data bank 1HYS
- FEM concentric spheres
- FEM cantilever
- Pressurized wind tunnel
- 3D CFD of Charleston harbor
- Quark propagators (QCD/LGT)
- FEM Ship section/detail
- Macroeconomic model
- 2D Markov model of epidemic
- Accelerator cavity design
- Motorola circuit simulation
- Web connectivity matrix
- Railways set cover Constraint matrix
SpMV: performance

SpMV Kernel

Gflops

- Xeon 5520 2.67 Ghz  25GB/s  (Intel MKL)
- Tesla M2050 144GB/s  (CuBLAS)
- Radeon 6970 830 Mhz 160 GB/s (Sliced ELL)
- Radeon 7970 925 Mhz 268 GB/s (Sliced ELL)

Heterogeneous parallel programming group ISR RAS
http://www.niisi.ru
SpMV: applications

- Krylov subspace iterative methods family
- CG – conjugate gradient, Bi-CGSTAB, …
- Preconditioned – incomplete inverse
Conclusions

The situation is not so bad, but there is no silver bullet

All the problems from that cursed PCI-E 1.x, 2.x, 3.x, x.x.x ...

OpenCL with command schedulers is a good choice for supercomputing on modern hybrid clusters (and probably future ones)
The results and prospects of: education

**2011:** Training course on heterogeneous programming in the framework of the standard OpenCL in Lomonosov Moscow state University

**2012:** Publication in the developer AMD forum: is the most popular theme for all time of the forum existence (about 6 years) http://devgurus.amd.com/message/1282591

**2013:** Russian translation of the standard OpenCL 1.2

**2013:** Heterogeneous programming methodology - 300 pages
The results and prospects: a program model

2013-...: Extending the package of tasks

2013-...: Extension of the model scheduler on a distributed system and the external input\output

2013: Partial porting NAS Parallel Benchmarks

2013-2014: The work under the project 12-01-33022 of the RFBR in 2013-2014 together with KIAM RAS

«Parallel high-precision algorithms on unstructured grids for the simulation of compressible turbulent flows on hybrid supercomputers»
The results and prospects: conference

2011: Parallel Computational Fluid Dynamics (ParCFD), Barcelona, Spain

A. V. Gorobets, S. A. Soukov, P. B. Bogdanov, A. O. Zheleznyakov and B. N. Chetverushkin, Extension with OpenCL of the two-level MPI+OpenMP parallelization for large-scale CFD simulations on heterogeneous systems, Parallel CFD 2011, Barcelona, Spain, 16-20 May 2011

2012: Parallel Computational Fluid Dynamics (ParCFD), Atlanta, USA

A.A. Efremov, P. B. Bogdanov, OpenCL mathematical software infrastructure for heterogeneous computing, Parallel CFD 2012, Atlanta, USA, 20-24 May 2012

2013: GPU technology conference (GTC), March 18-21, San Jose, California, USA

A.A. Efremov, P. B. Bogdanov, Programming Infrastructure for Heterogeneous Computing Based on OpenCL and its Applications

2013: Exascale Application and Software Conference (EASC), April 9-11, Edinburg, Scotland, UK

A.A. Efremov, P. B. Bogdanov, Programming infrastructure for heterogeneous computing based on OpenCL and its applications

2013: Parallel Computational Fluid Dynamics (ParCFD), May 20-24, Changsha, Hunan, China

2013: AMD 2013 Developer Summit, November 11-14, San Jose, California, USA
Thank you for watching!