Автоматизация разработки параллельных программ для гетерогенных вычислительных кластеров

В.А. Бахтин, В.А. Крюков,
А.С. Колганов, Н.В. Поддерюгина, М.Н. Притула

bakhtin@keldysh.ru

Федеральное государственное учреждение
ФЕДЕРАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЦЕНТР
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ ИМ. М.В. КЕЛДЫША
Российской академии наук

План доклада

- DVMH-модель параллельного программирования
- Преимущества DVMH-модели
- Новые возможности DVM-системы:
  - новая версия компилятора и языка C-DVMH;
  - параллельный ввод-вывод;
  - новые возможности для функциональной отладки и анализа эффективности параллельных программ.
- Система САПФОР

© ФИЦ ИПМ им. М.В. Келдыша РАН

TOP 500 The List.

PRESENTED BY

ICL UT INNOVATIVE COMPUTING LABORATORY THE UNIVERSITY of TENNESSEE

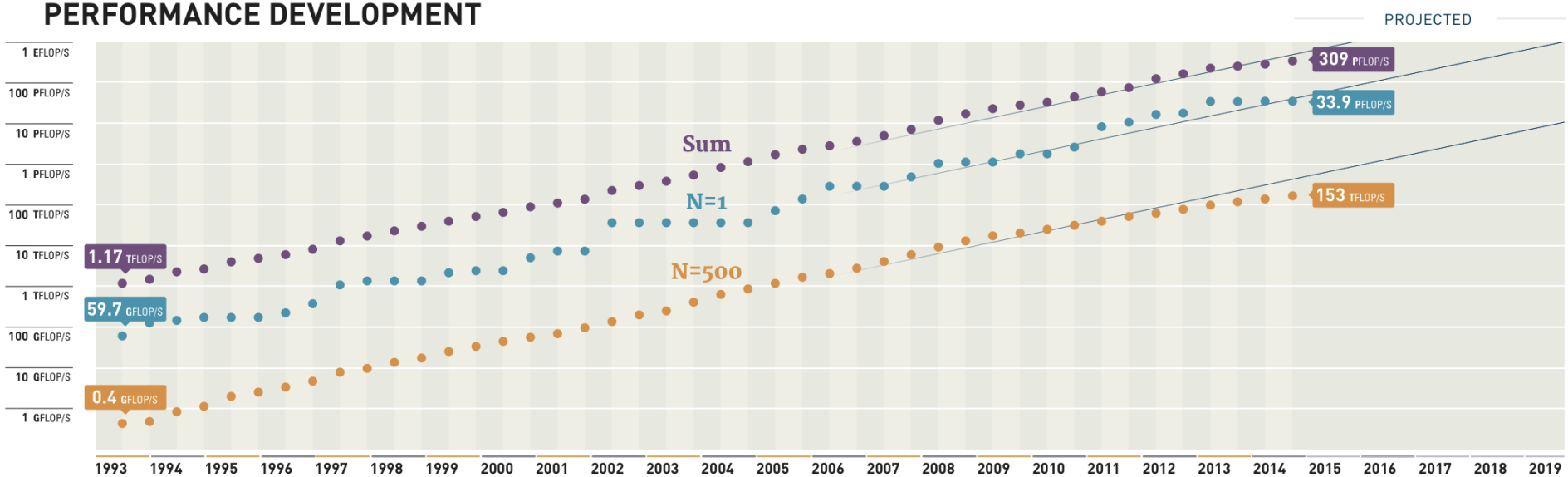
BERKELEY LAB Lawrence Berkeley National Laboratory

PROMETEUS PROFESSOR MEUER TECHNOLOGIEBERATUNG

FIND OUT MORE AT top500.org

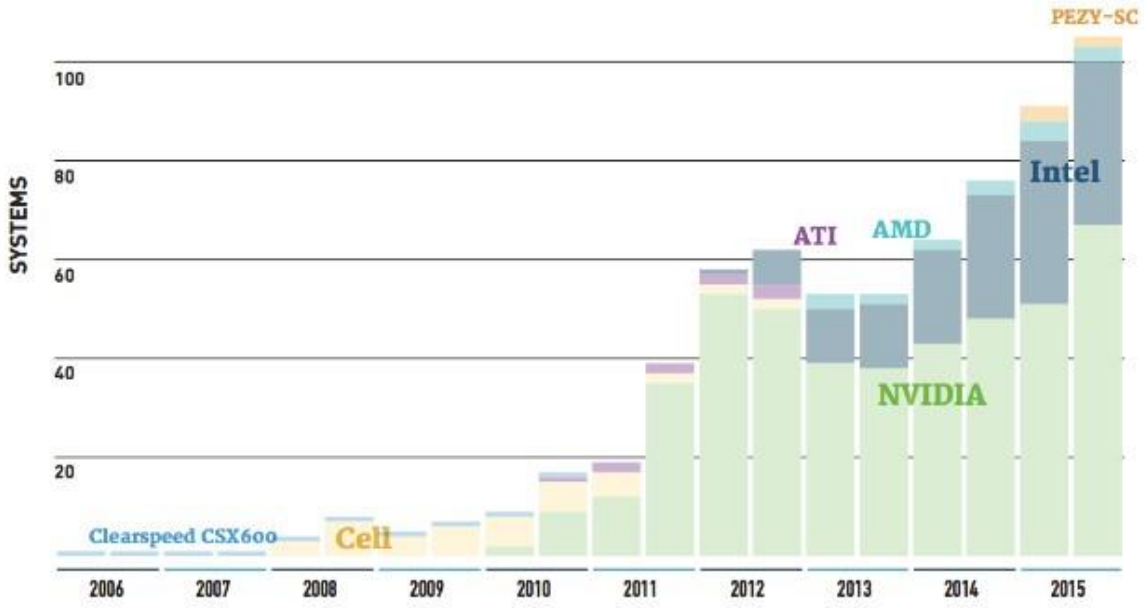
	NAME	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Tianhe-2 (Milkyway-2)	NUDT, Intel Ivy Bridge (12C, 2.2 GHz) & Xeon Phi (57C, 1.1 GHz), Custom interconnect	NSCC Guangzhou	China	3,120,000	33.9	17.8
2	Titan	Cray XK7, Opteron 6274 (16C 2.2 GHz) + Nvidia Kepler GPU, Custom interconnect	DOE/SC/ORNL	USA	560,640	17.6	8.2
3	Sequoia	IBM BlueGene/Q, Power BQC (16C 1.60 GHz), Custom interconnect	DOE/NNSA/LLNL	USA	1,572,864	17.2	7.9
4	K computer	Fujitsu SPARC64 VIIIfx (8C, 2.0GHz), Custom interconnect	RIKEN AICS	Japan	705,024	10.5	12.7
5	Mira	IBM BlueGene/Q, Power BQC (16C, 1.60 GHz), Custom interconnect	DOE/SC/ANL	USA	786,432	8.59	3.95

PERFORMANCE DEVELOPMENT



Top 500. Использование ускорителей

Accelerator	Count	System Share (%)	Rmax (GFlops)	Rpeak (GFlops)	Cores
Nvidia Kepler	52	10.4	59,907,300	90,909,313	1,529,135
Intel Xeon Phi	29	5.8	57,080,027	89,469,693	4,903,752
Nvidia Fermi	14	2.8	8,891,651	18,165,047	691,712
Hybrid	4	0.8	8,142,240	12,403,928	681,400
ATI Radeon	3	0.6	1,148,600	2,280,349	94,304
PEZY-SC	2	0.4	812,196	1,908,851	1,444,736



Алгоритм Якоби в модели DVMH

```
          PROGRAM    JACOBY_DVMH
          PARAMETER    (L=4096,  ITMAX=100)
          REAL     A(L,L), B(L,L)
!DVM$     DISTRIBUTE    ( BLOCK,   BLOCK)   ::   A
!DVM$     ALIGN  B(I,J)  WITH  A(I,J)
          PRINT *,  '**********  TEST_JACOBI   **********'
          DO IT  =  1,  ITMAX
!DVM$        REGION INOUT(A,B)
!DVM$           PARALLEL (J,I) ON  A(I,  J)
                DO  J  =  2, L-1
                     DO  I  =  2, L-1
                          A(I, J)  =  B(I, J)
                     ENDDO
                ENDDO
!DVM$           PARALLEL (J,I) ON B(I,  J), SHADOW_RENEW (A)
                DO  J = 2,  L-1
                      DO  I = 2,  L-1
                           B(I, J) =  (A(I-1, J) + A(I, J-1) +  A(I+1, J) + A(I, J+1)) / 4
                      ENDDO
                ENDDO
!DVM$        END REGION
             ENDDO
!DVM$     IO_MODE(LOCAL, ASYNCHRONOUS)
          OPEN(3, FILE='JAC.DAT', ACCESS='STREAM')
!DVM$     GET_ACTUAL(B)
          WRITE(3, *) B
          CLOSE(3)
          END
```

© ФИЦ ИПМ им. М.В. Келдыша РАН

Спецификации параллельного выполнения программы

- Распределение элементов массива между процессорами
- Распределение витков цикла между процессорами
- Спецификация параллельно выполняющихся секций программы (параллельных задач) и отображение их на процессоры
- Организация эффективного доступа к удаленным (расположенным на других процессорах/ускорителях) данным

© ФИЦ ИПМ им. М.В. Келдыша РАН

Спецификации параллельного выполнения программы

- Организация эффективного выполнения редукционных операций - глобальных операций с расположенными на различных процессорах/ускорителях данными (таких, как их суммирование или нахождение их максимального или минимального значения)
- Определение фрагментов программы (регионов) для возможного выполнения на ускорителях
- Управление перемещением данных между памятью ЦПУ и памятью ускорителей
- Управление параллельным вводом-выводом

© ФИЦ ИПМ им. М.В. Келдыша РАН

Состав DVM-системы

Система состоит из следующих компонент:

- Компилятор Fortran-DVMH
- Компилятор C-DVMH
- Библиотека поддержки LIB-DVMH
- DVMH-отладчик
- Анализатор производительности DVMH-программ

© ФИЦ ИПМ им. М.В. Келдыша РАН

XcalableACC=XcalableMP+OpenACC

RIKEN Advanced Institute for Computational Science
University of Tsukuba
The University of Tokyo
http://www.xcalablemp.org/download/publication/2014/WACCPD.pdf

Omni XMP Compiler http://omni-compiler.org

```
FILE *f;
#pragma  xmp nodes p(N, N)
#pragma  xmp template t(0:L−1,  0:L−1)
#pragma  xmp distribute t(block,  block)
#pragma  xmp align [i][j] with  t(i,j) :: A,B
#pragma  xmp shadow  A[1:1]
double A[L][L];
double B[L][L];
```

Алгоритм Якоби в модели XACC

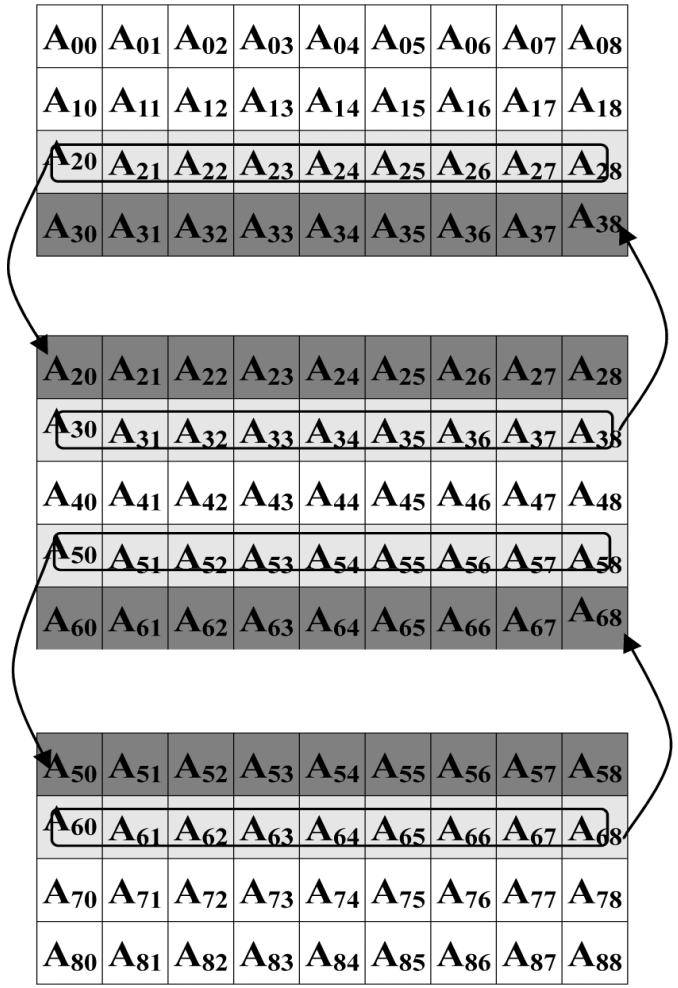
```
int main(int argc, char *argv[]) {
  #pragma  acc  data copy(B) copyin(A)
  {
   for(int it = 0; it < ITMAX; it++) {
          #pragma  xmp  loop  (i,j)  on  t(i,j)
          #pragma  acc  parallel  loop  collapse(2)
         for (int i = 1; I < L - 1; i++)
           for (int j = 1; j < L-1; j++)  A[i][j] = B[i][j];
        #pragma  xmp  reflect (A) acc
        #pragma  xmp  loop  (i,j)  on  t(i,j)
        #pragma  acc  parallel  loop  collapse(2)
         for (int i = 1; I < L - 1; i++)
           for (int j = 1; j < L - 1; j++)
               B[i][j] = (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]) / 4.;
      }
   }
  }
return 0;
}
```

Алгоритм Якоби в модели ХАСС

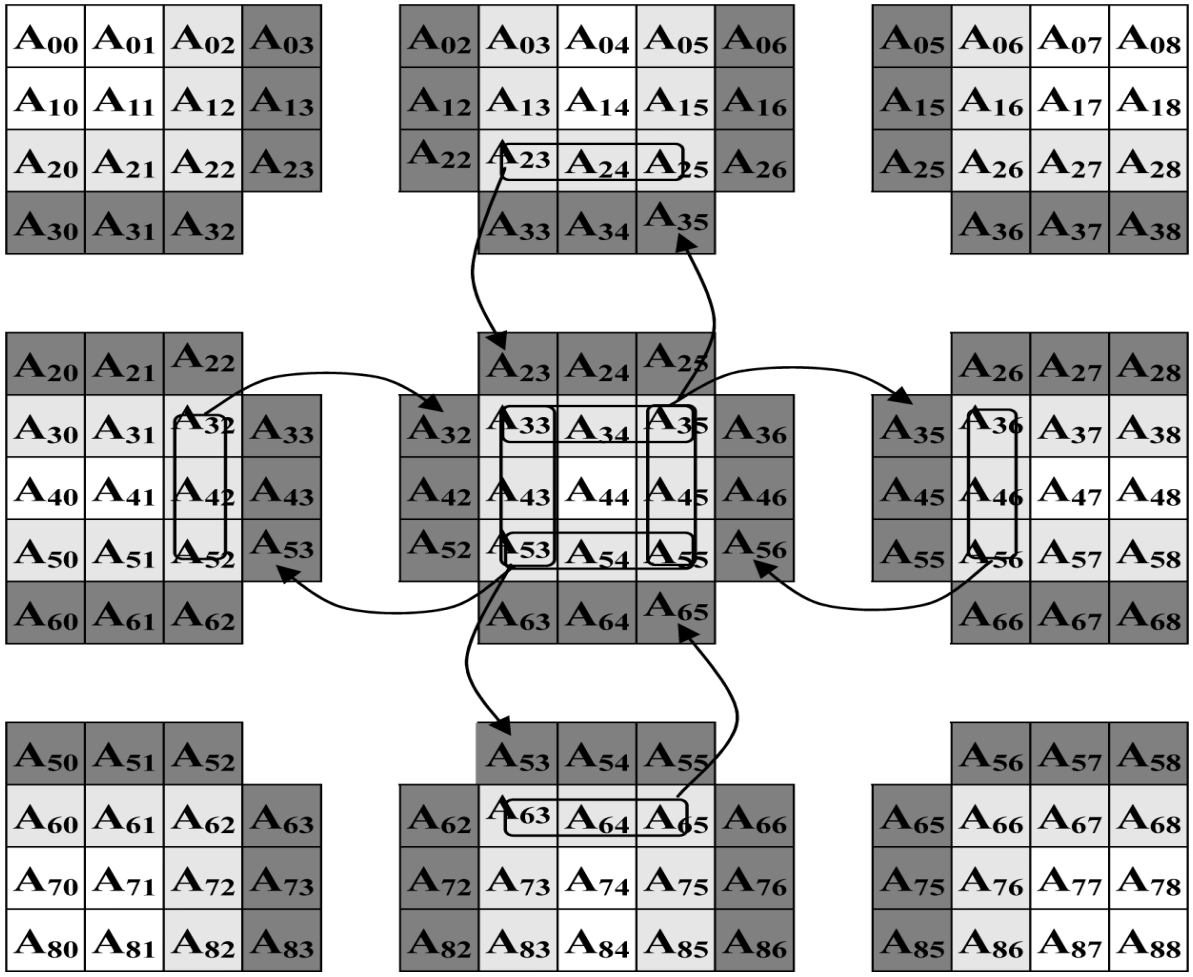
© ФИЦ ИПМ им. М.В. Келдыша РАН

Распределение массива

./dvm run 3 1 jac



./dvm run 3 3 jac



© ФИЦ ИПМ им. М.В. Келдыша РАН

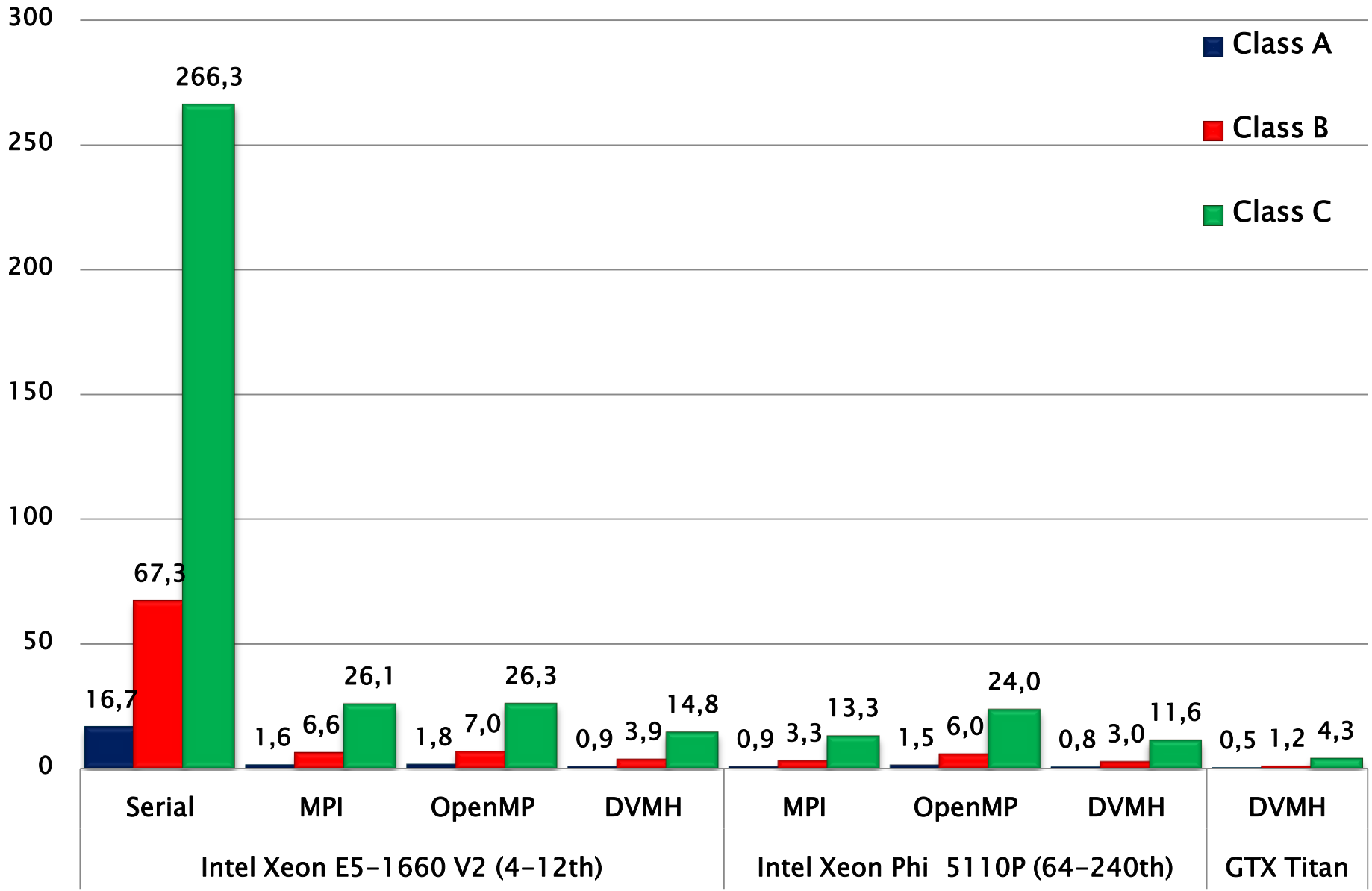
Распараллеливание тестов NAS

- **EP** - генерация пар случайных чисел Гаусса
- **MG** - приближенное решение трехмерного уравнения Пуассона. Метод MultiGrid
- **BT** - 3D Навье-Стокс, блочная трехдиагональная схема. Метод переменных направлений
- **LU** - 3D Навье-Стокс. Метод последовательной верхней релаксации
- **SP** - 3D Навье-Стокс. Скалярная пятидиагональная схема. Beam-Warning approximate factorization

Подробные результаты исследования эффективности доступны на сайте http://www.dvm-system.org

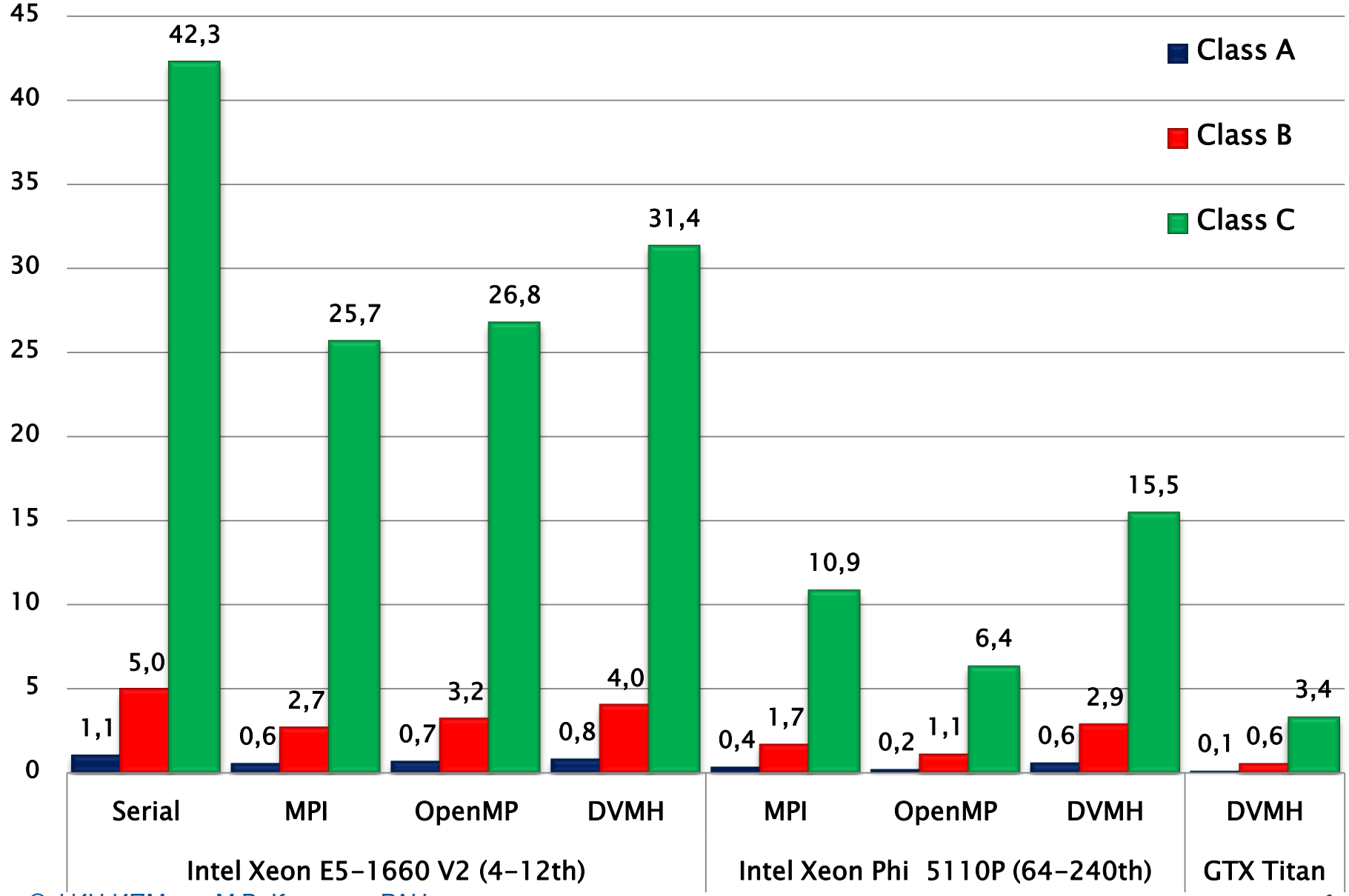
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения теста EP, в секундах



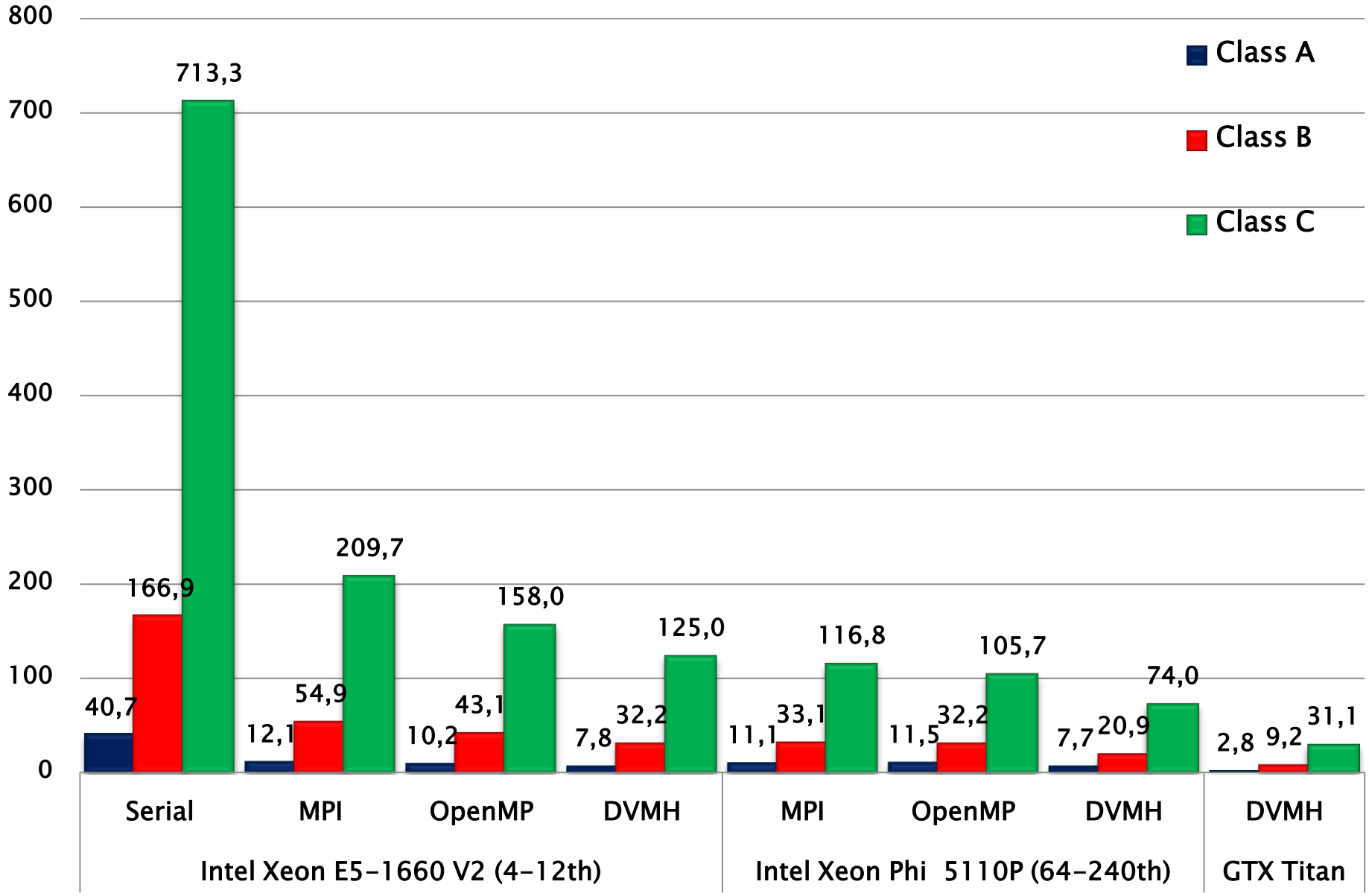
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения теста MG, в секундах



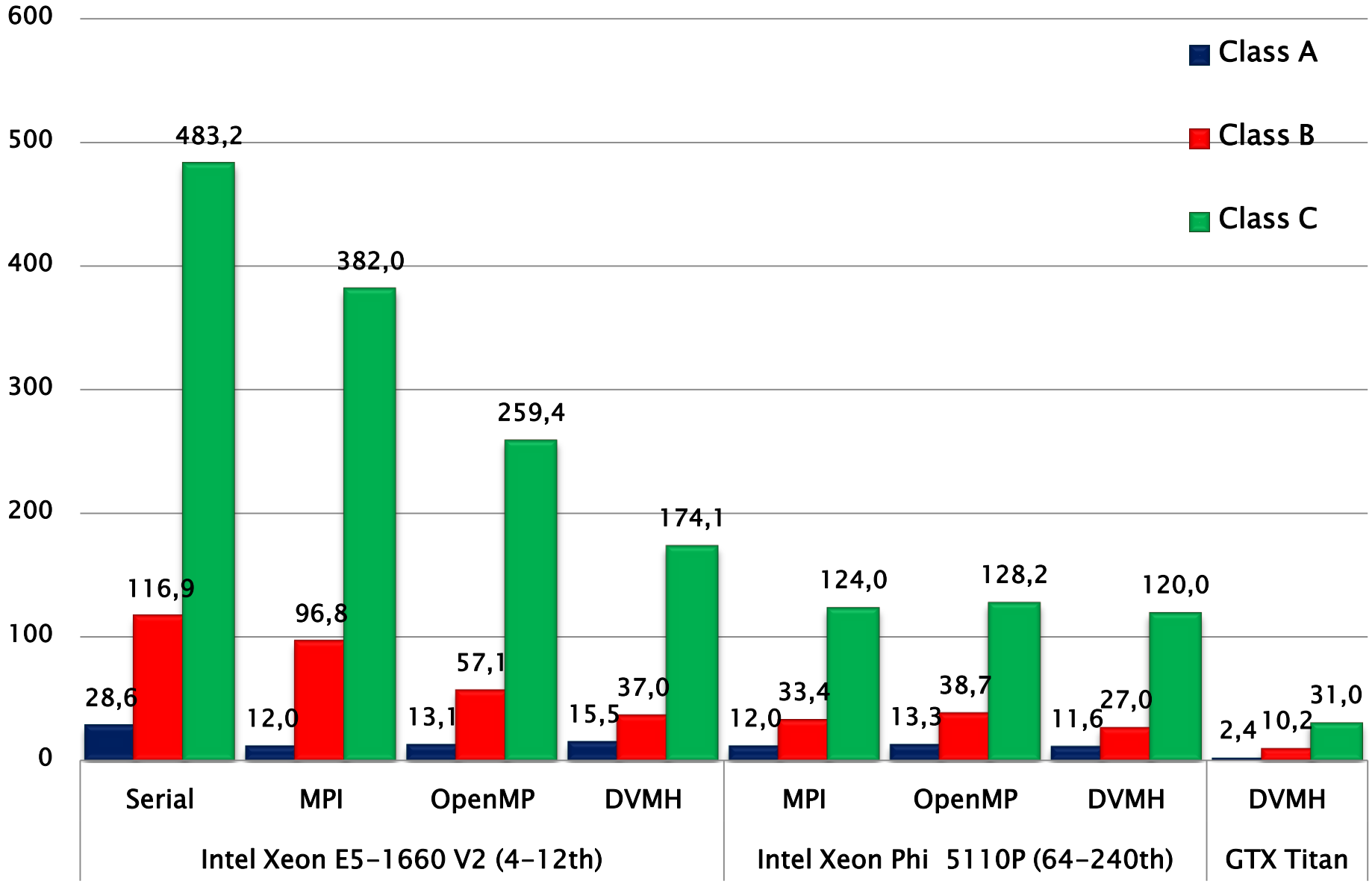
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения теста BT, в секундах



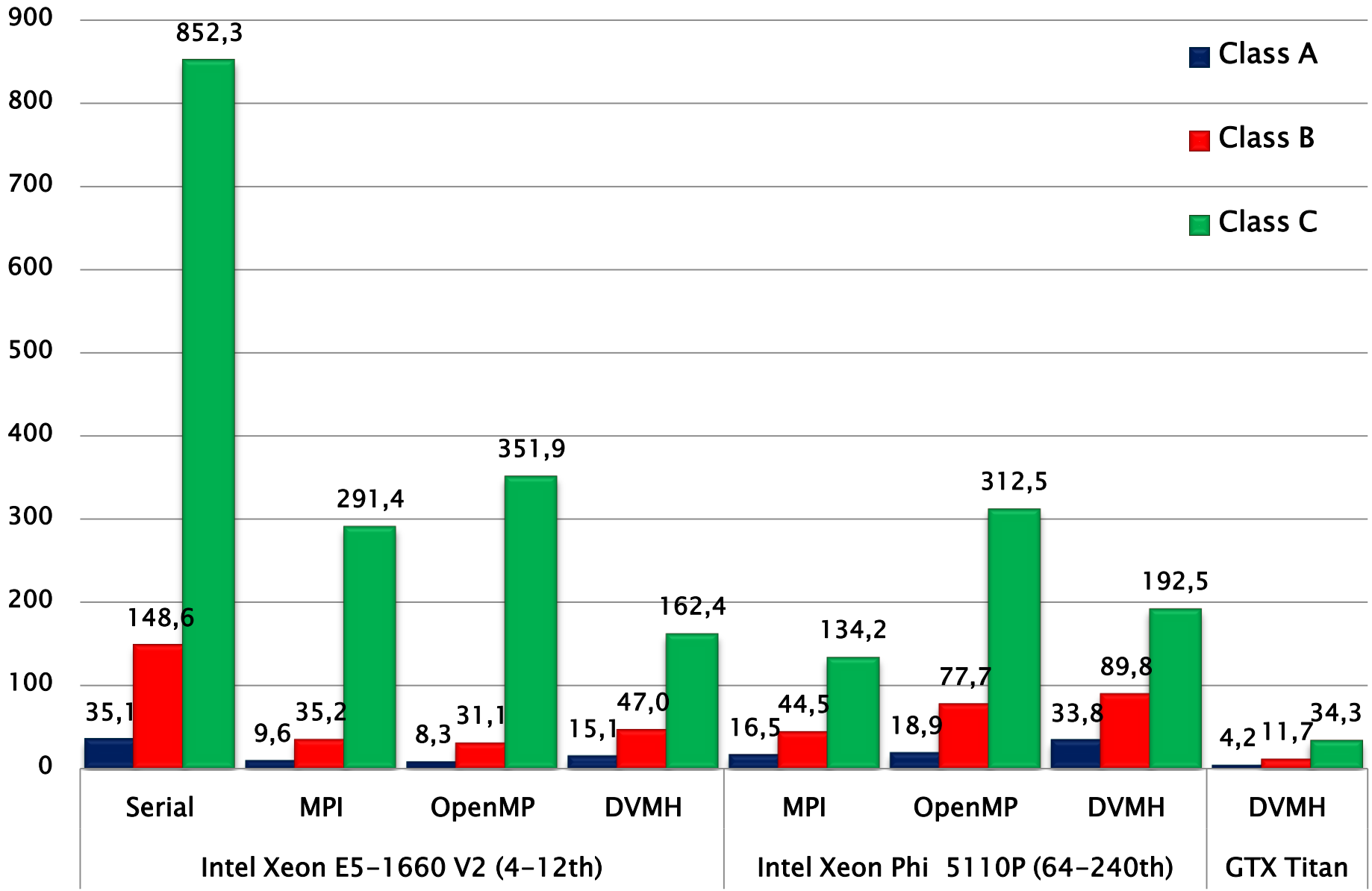
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения теста SP, в секундах



© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения теста LU, в секундах



© ФИЦ ИПМ им. М.В. Келдыша РАН

Тесты NAS для GPU

- OpenCL-версии

  Center for Manycore Programming at

  Seoul National University (SNU NPB Suite)
- CUDA-версии

  Chemnitz University of Technology (BT, LU, SP)

  Laercio Lima Pilla from the Federal University

  of Rio Grande do Sul (EP, CG, FT)
- OpenACC-версии

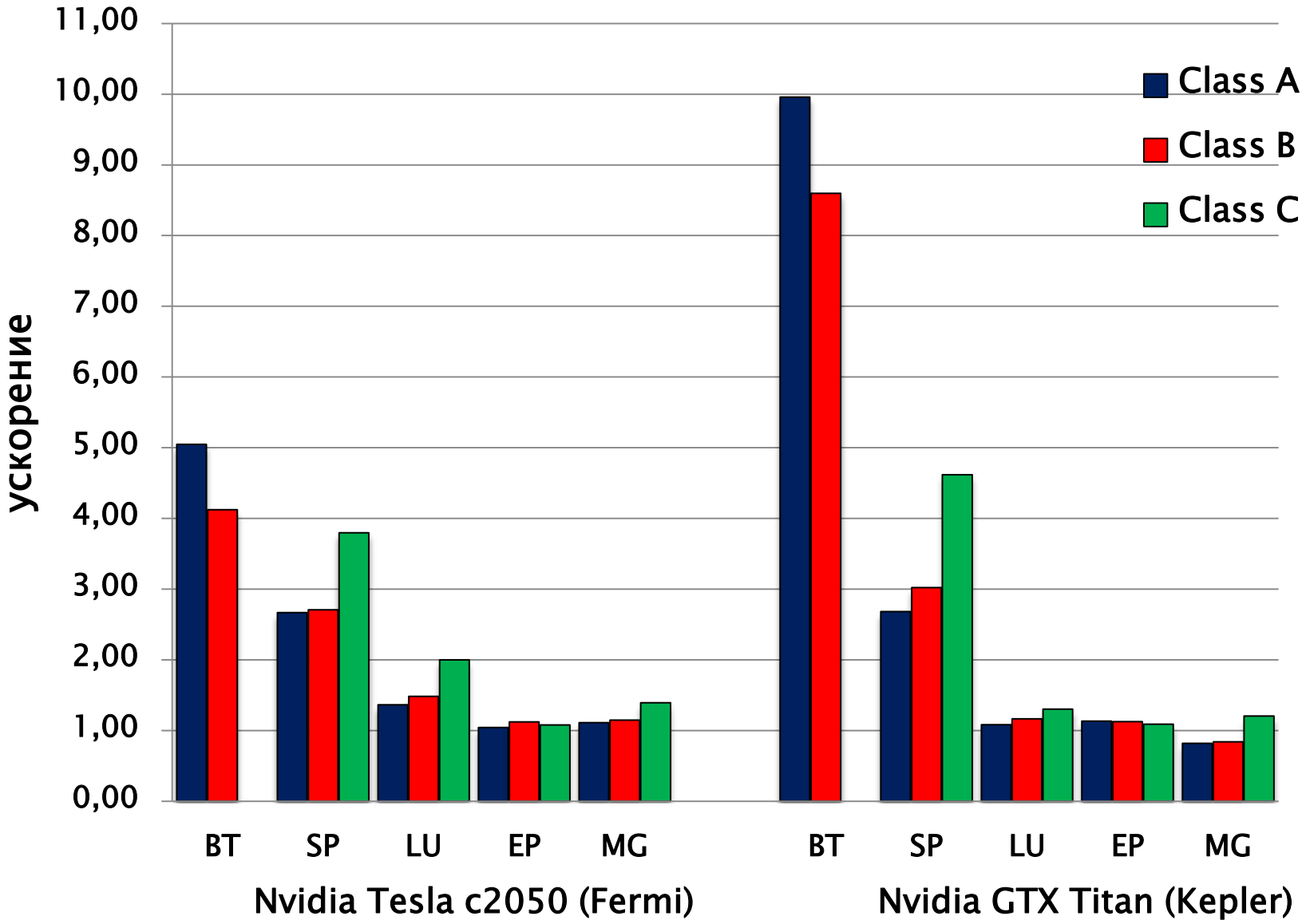
  HPC Tools group, Department of Computer

  Science University of Houston

  Rengan Xu, Barbara Chapman et al.

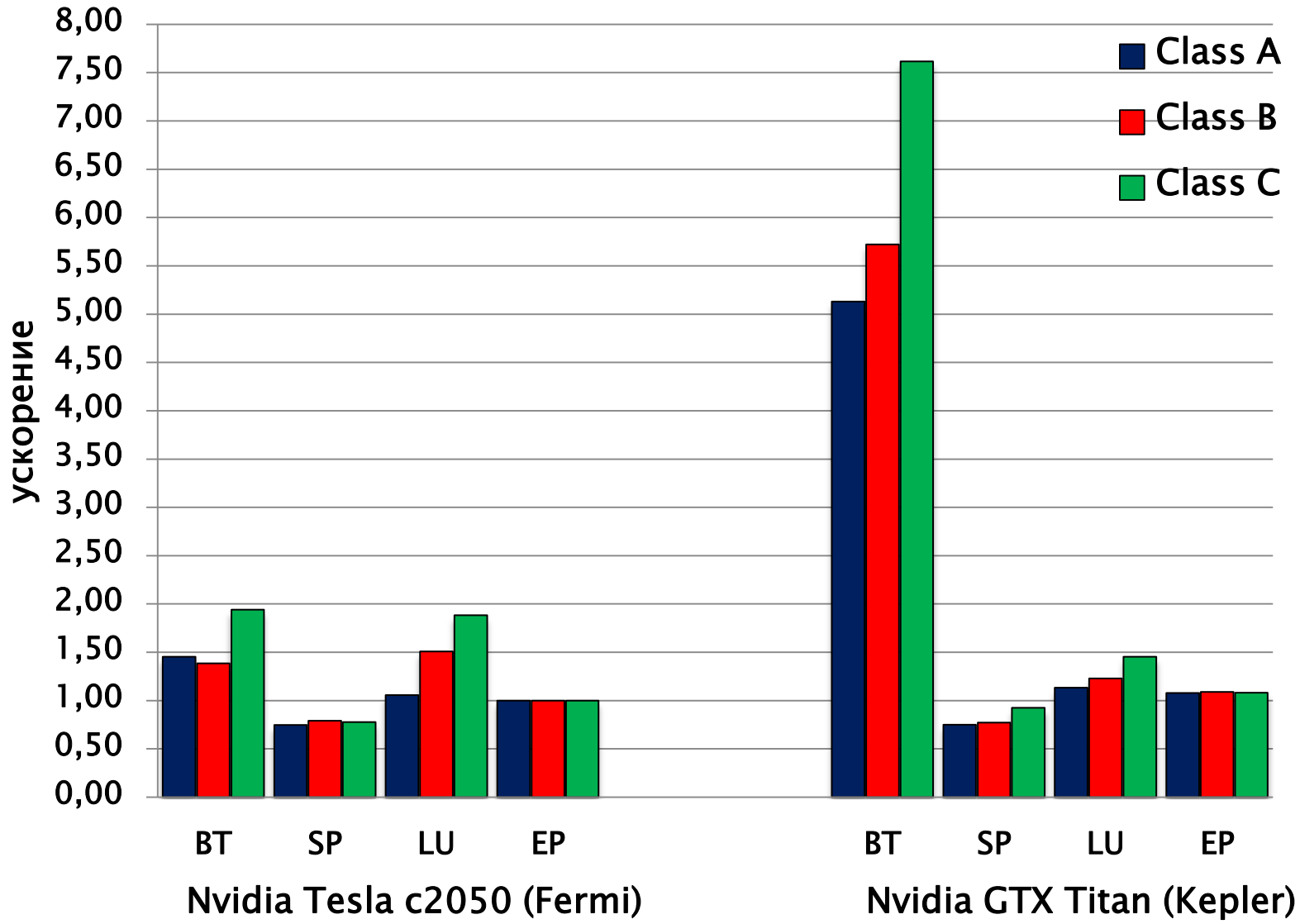
© ФИЦ ИПМ им. М.В. Келдыша РАН

Ускорение выполнения DVMH-версий тестов NAS NPB по сравнению с OpenCL-версиями



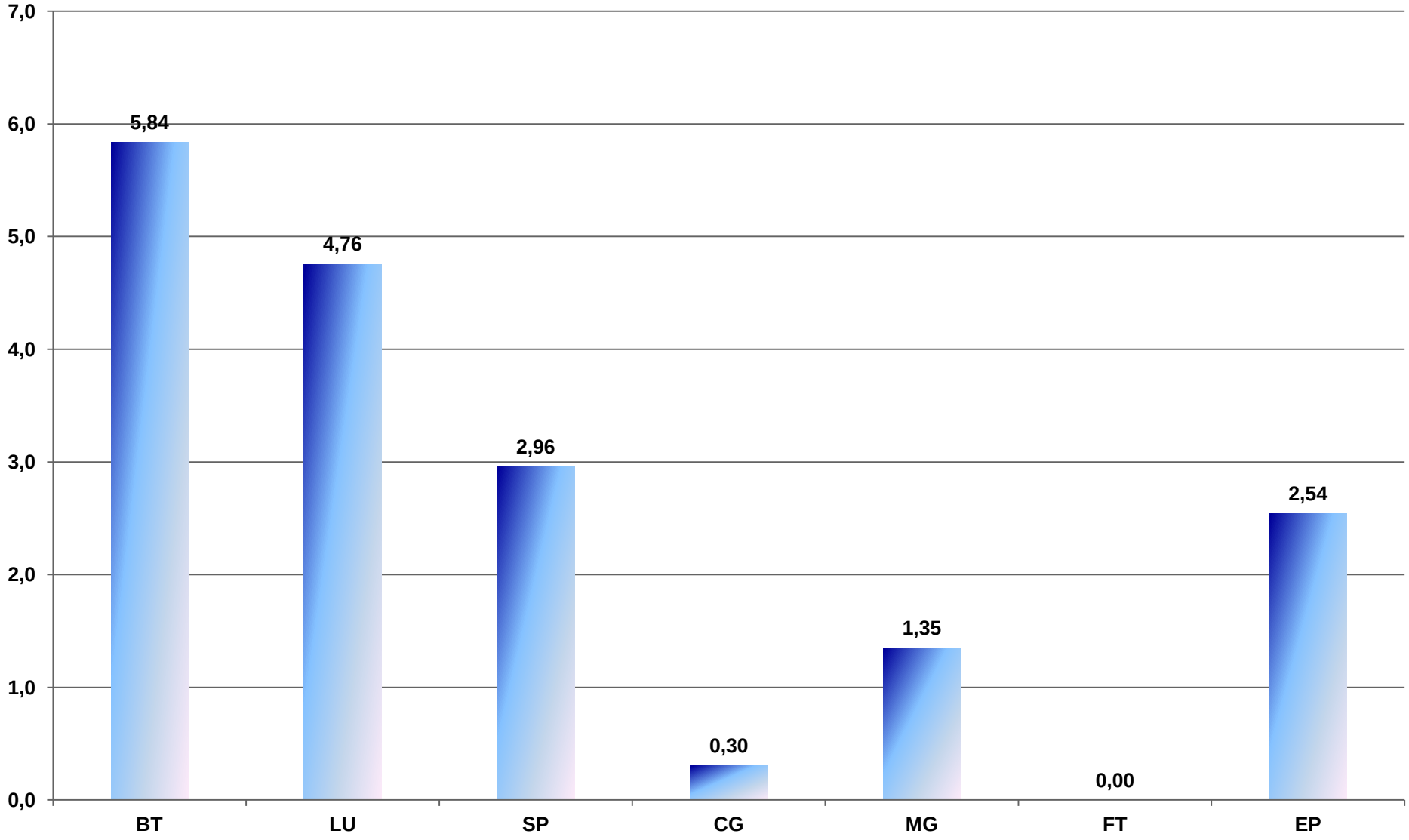
© ФИЦ ИПМ им. М.В. Келдыша РАН

Ускорение выполнения DVMH-версий тестов NAS NPB по сравнению с CUDA-версиями



© ФИЦ ИПМ им. М.В. Келдыша РАН

Ускорение выполнения DVMH-версий тестов NAS NPB по сравнению с OpenACC версиями (класс C)



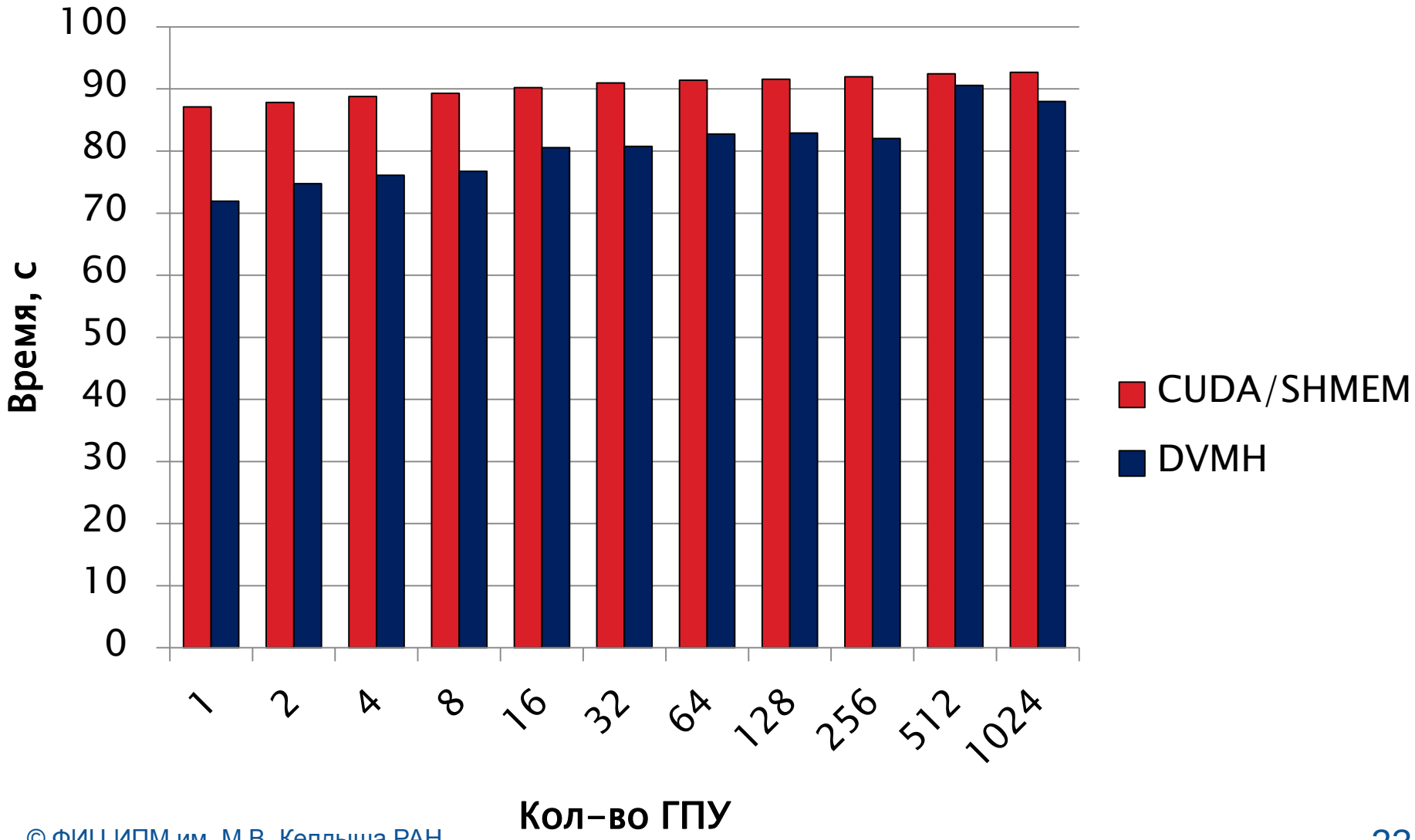
© ФИЦ ИПМ им. М.В. Келдыша РАН

Применение DVMH для приложений

- **Контейнер** - моделирование течения вязкой тяжелой жидкости под действием силы тяжести в прямоугольном контейнере с открытой верхней стенкой и отверстием в одной из боковых стенок в трехмерной постановке (Fortran-DVMH)
- **POLINOM** - расчет коэффициентов полиномиальной реконструкции (C-DVMH)
- **FLUXES -** расчет конвективных потоков через внутренние грани расчетных ячеек (C-DVMH)
- **Композит –** композиционная модель многокомпонентной фильтрации при разработке месторождений нефти и газа (Fortran-DVMH)

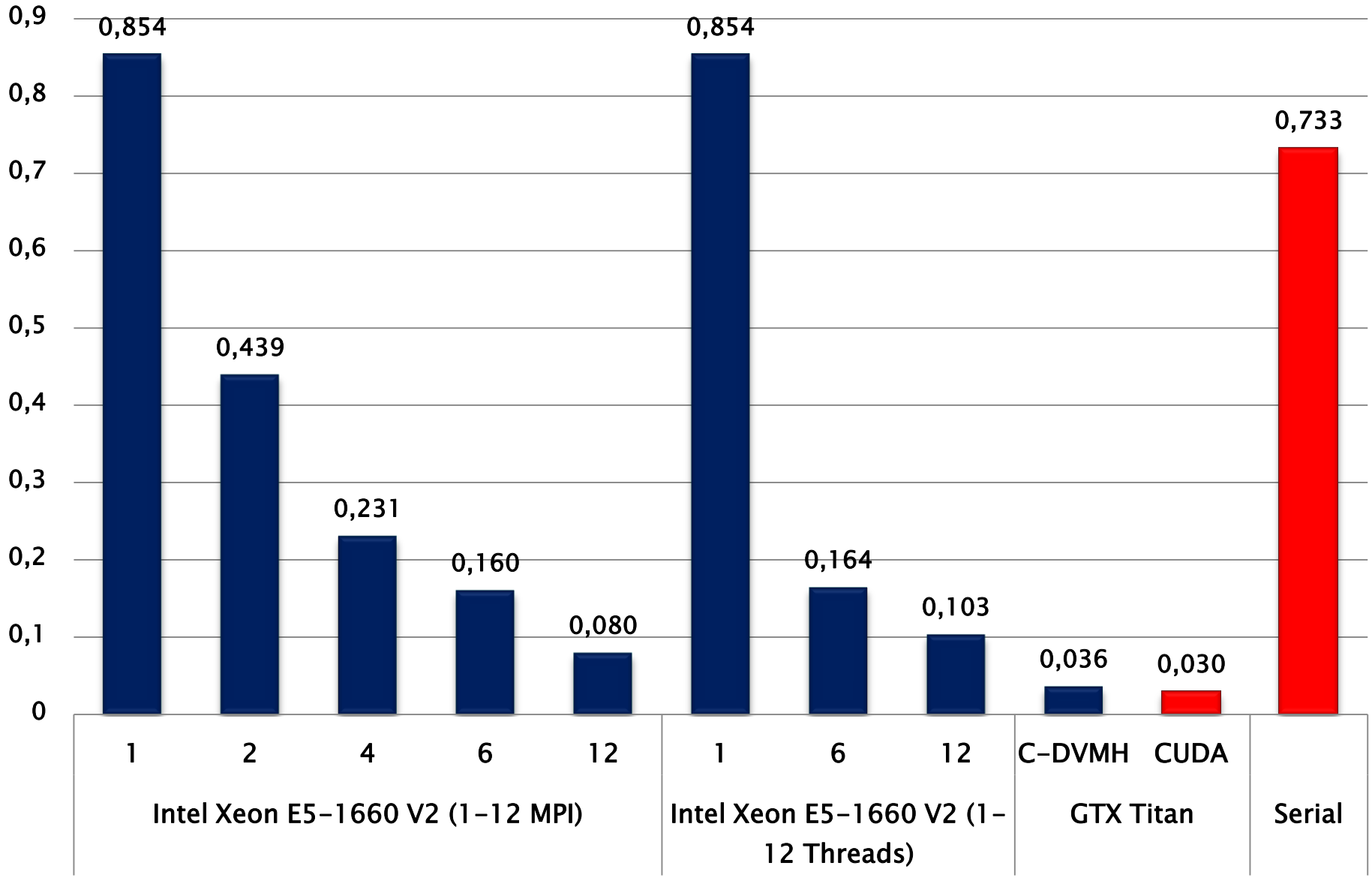
© ФИЦ ИПМ им. М.В. Келдыша РАН

Программа Контейнер. Сравнение DVMH vs C+SHMEM+CUDA



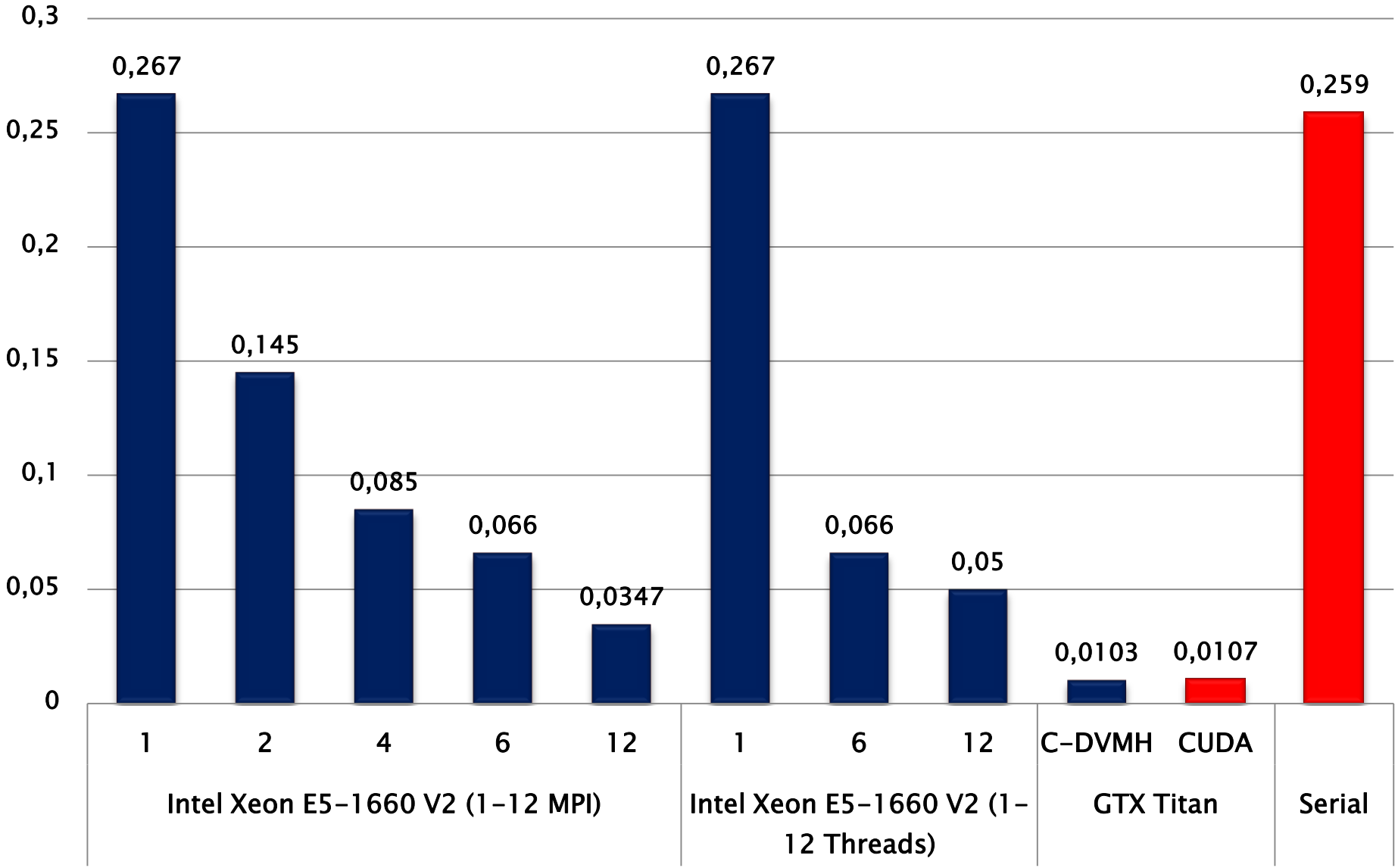
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения программы POLINOM



© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения программы FLUXES



© ФИЦ ИПМ им. М.В. Келдыша РАН

Методы динамической настройки DVMH-программ

- Отображение массивов и циклов на узлы кластера с учетом их производительности
- Отображение массивов и циклов на устройства узла с учетом их производительности
- Трансформация массивов
- Использование динамической компиляции

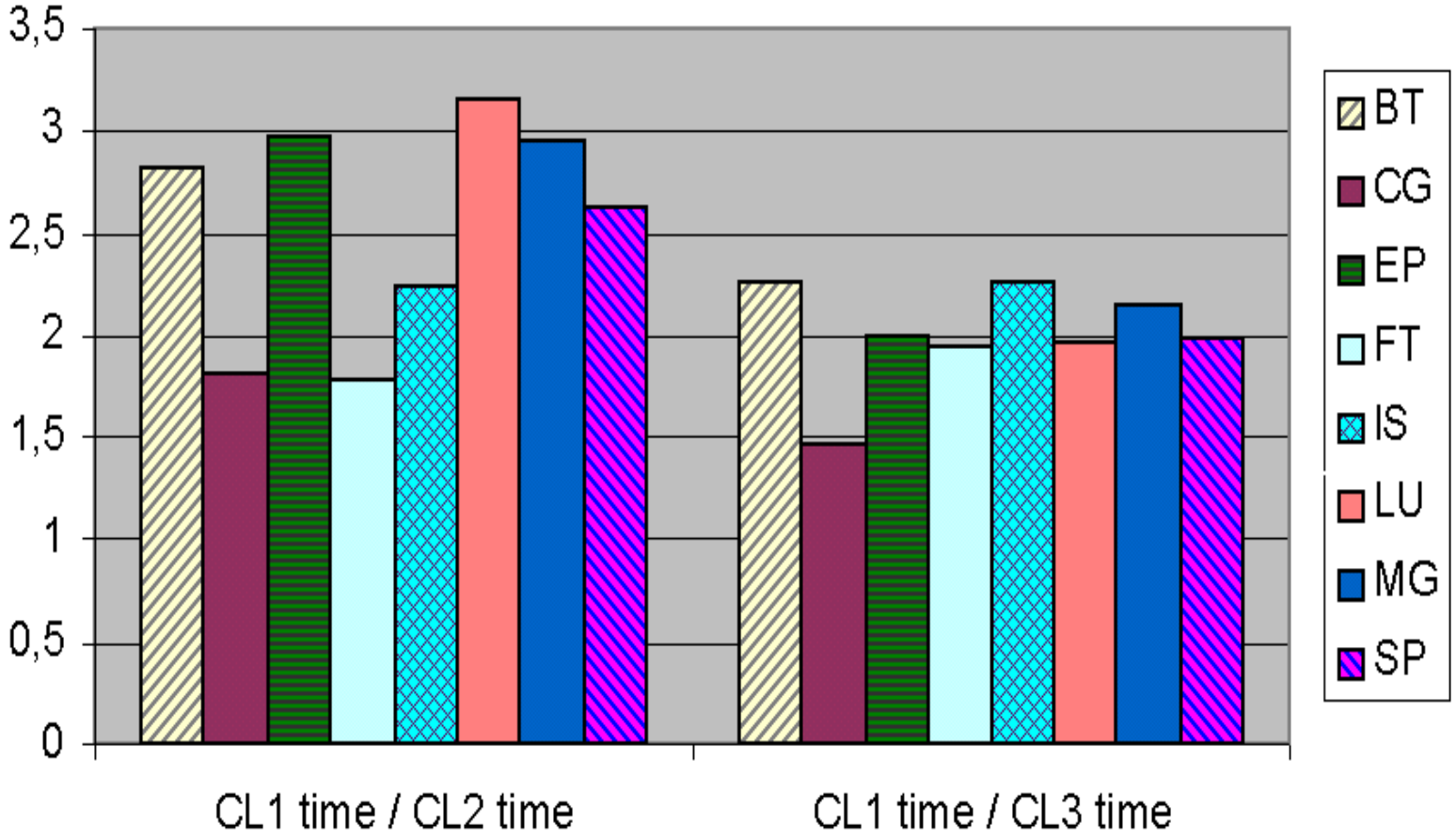
© ФИЦ ИПМ им. М.В. Келдыша РАН

Методы динамической настройки DVMH-программ

- Отображение массивов и циклов на узлы кластера с учетом их производительности
- Отображение массивов и циклов на устройства узла с учетом их производительности
- Трансформация массивов
- Использование динамической компиляции

© ФИЦ ИПМ им. М.В. Келдыша РАН

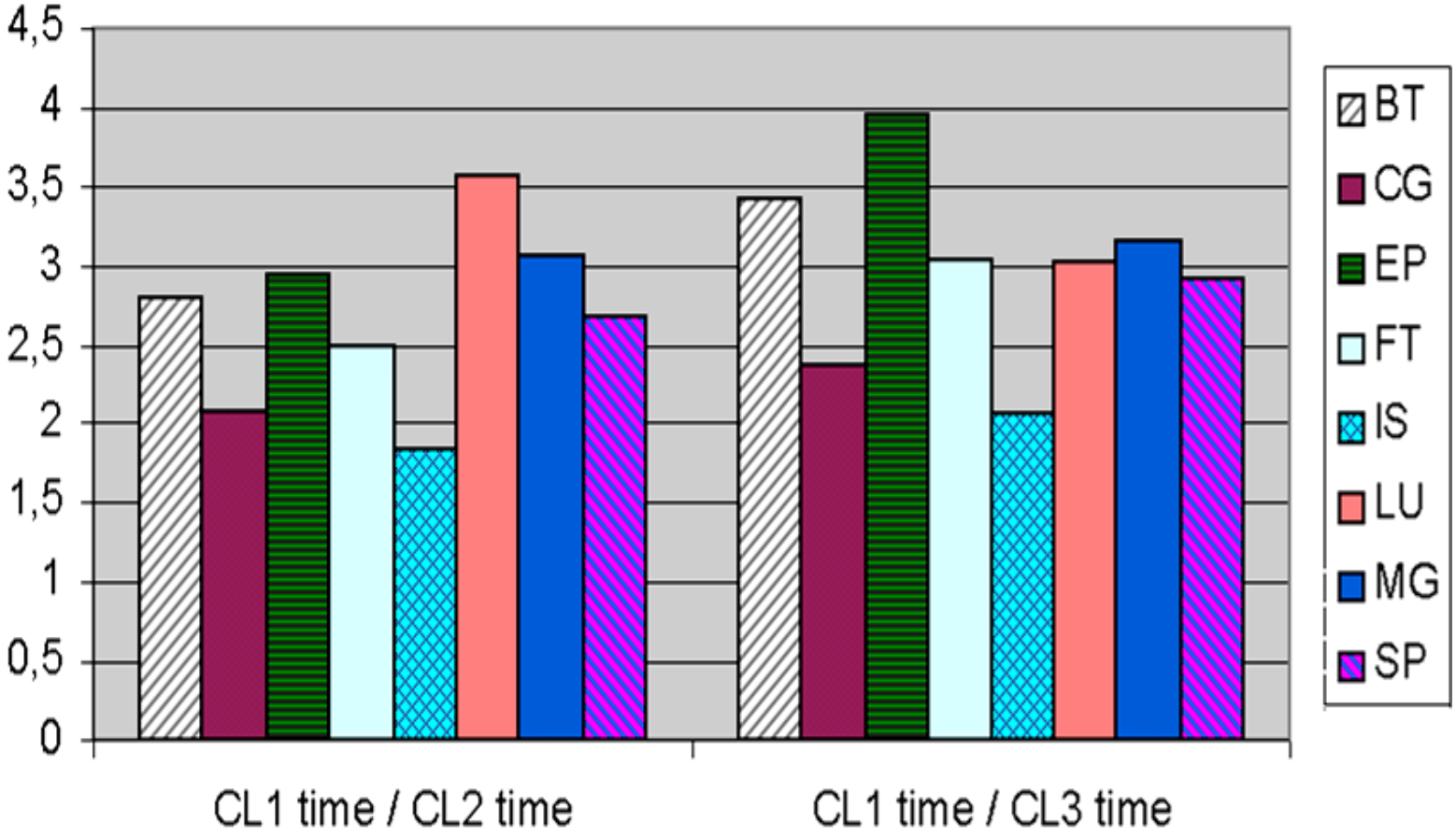
Ускорение выполнения MPI-версий тестов NAS NPB на неоднородном кластере



CL1 – 128 процессоров со скоростью выполнения P,
CL2 – 128 процессоров со скоростью выполнения 3P,
CL3 – 128 процессоров со скоростью выполнения P и 128 процессоров со скоростью выполнения 3P.

© ФИЦ ИПМ им. М.В. Келдыша РАН

Ускорение выполнения DVMH-версий тестов NAS NPB на неоднородном кластере



CL1 – 128 процессоров со скоростью выполнения P,
CL2 – 128 процессоров со скоростью выполнения 3P,
CL3 – 128 процессоров со скоростью выполнения P и 128 процессоров со скоростью выполнения 3P.

© ФИЦ ИПМ им. М.В. Келдыша РАН

Методы динамической настройки DVMH-программ

- Отображение массивов и циклов на узлы кластера с учетом их производительности
- Отображение массивов и циклов на устройства узла с учетом их производительности
- Трансформация массивов
- Использование динамической компиляции

© ФИЦ ИПМ им. М.В. Келдыша РАН

Режимы распределения данных и вычислений по вычислительным устройствам

- Простой статический режим
- Динамический режим с подбором схемы распределения
- Динамический режим с использованием подобранной схемы распределения

© ФИЦ ИПМ им. М.В. Келдыша РАН

Простой статический режим

- В каждом регионе одинаковое распределение по устройствам
- Задается пользователем в виде вектора относительных производительностей устройств

  export DVMH_CPU_PERF='0.35016‘

  export DVMH_CUDAS_PERF='0.64984‘
- Не учитывается различное соотношение производительности в разных регионах

© ФИЦ ИПМ им. М.В. Келдыша РАН

Динамический режим

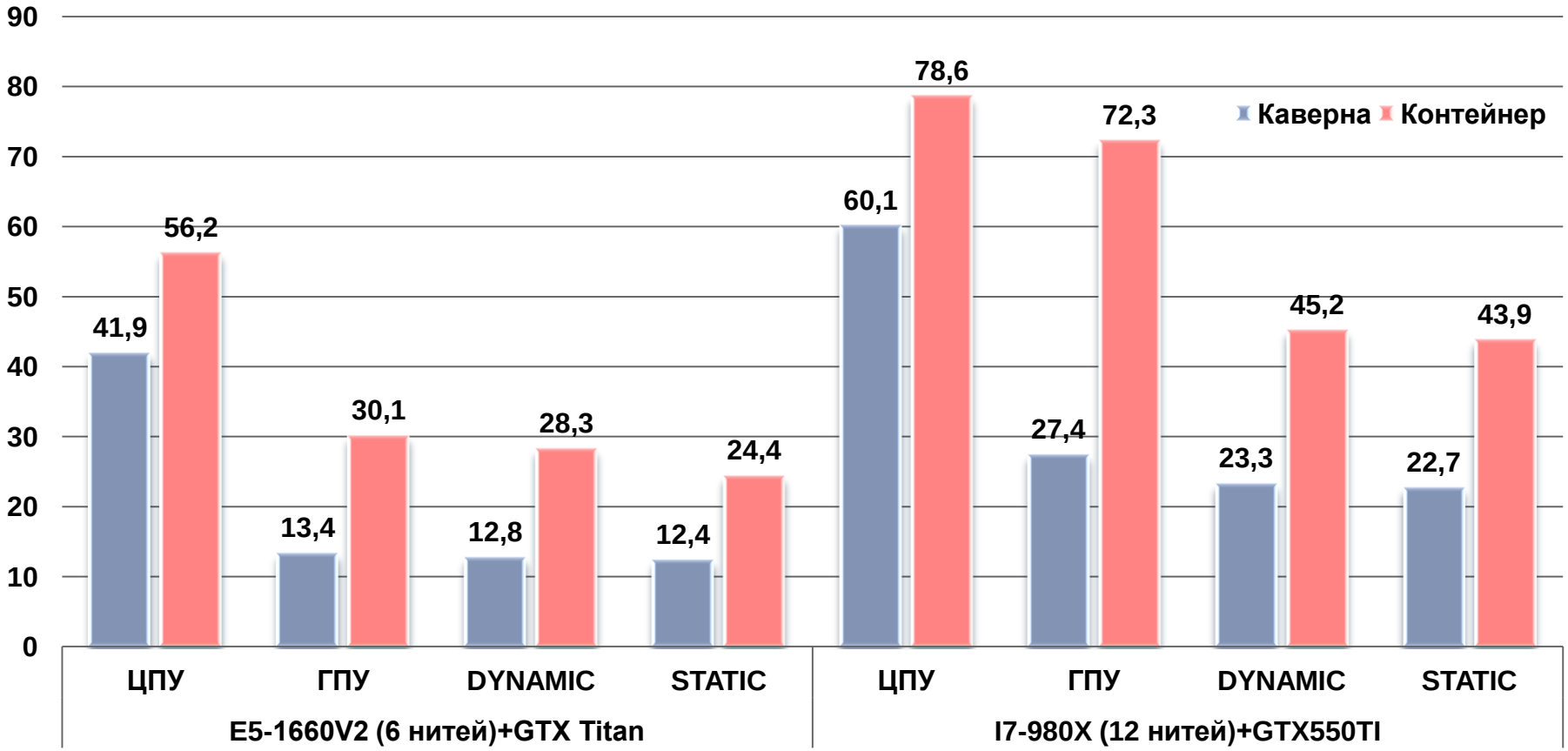
- Для регионов и параллельных циклов строится зависимость их времени работы от распределения
- Строится зависимость затрат на перемещения данных от распределения
- Производится планирование выполнения с целью минимизации общего времени
- При завершении – вывод результатов планирования

© ФИЦ ИПМ им. М.В. Келдыша РАН

Зависимость производительности от объема вычислений

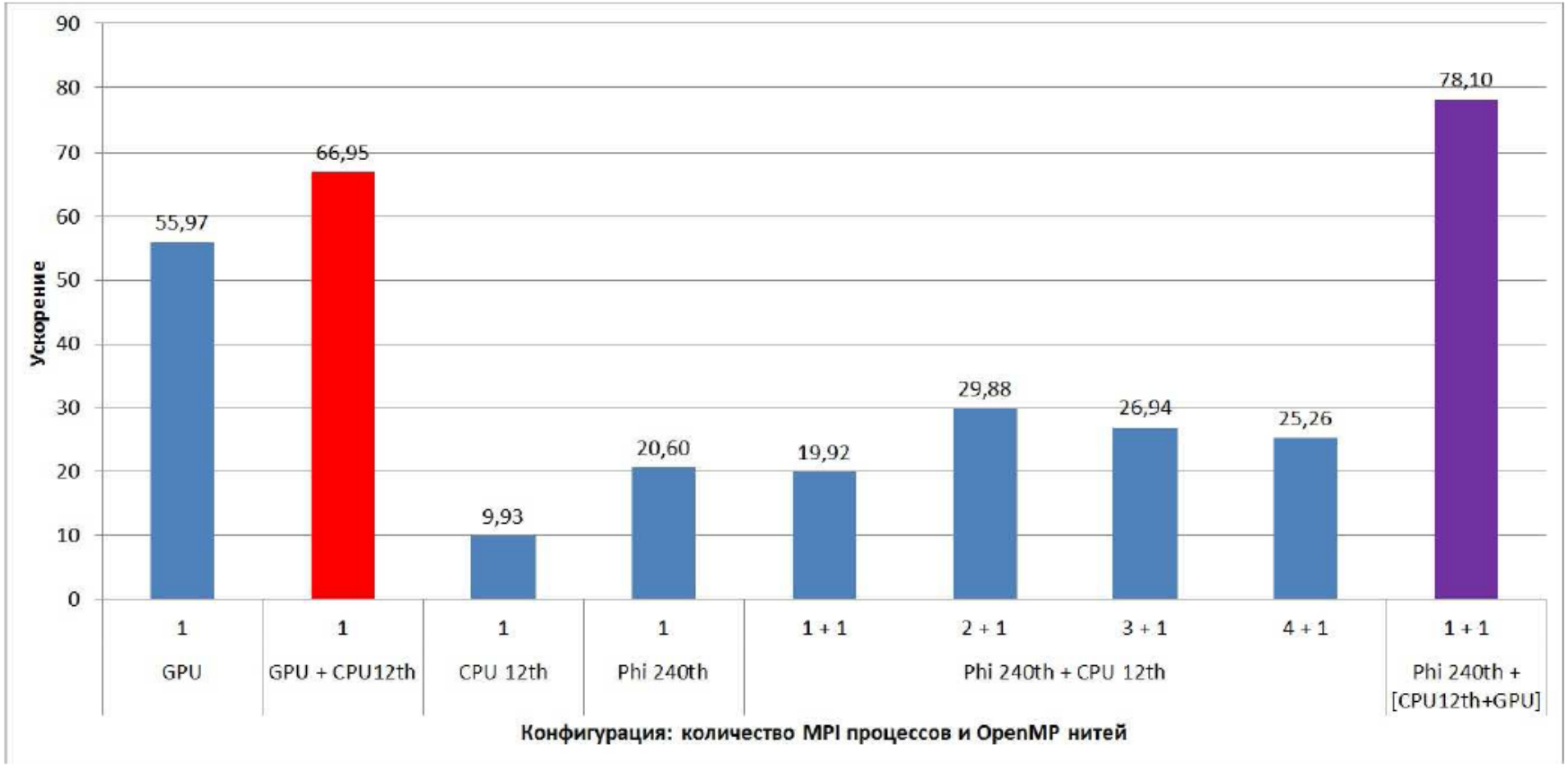
```
Performance statistics for parallel loop at cavity.fdv(601):
   Device #0:
      Handler #0:
         Best parameters: threads=11
         Table function (iterations => performance):
                 894400 => 2.21928e+08
                  1e+06 => 2.20961e+08
               2.56e+06 => 2.22738e+08
   Device #1:
      Handler #0:
         Best parameters: thread-block=(32,8,1)
         Table function (iterations => performance):
               1.56e+06 => 6.3996e+08
             1.6656e+06 => 6.44597e+08
               2.56e+06 => 6.74815e+08
Simple dynamic run performances:
DVMH_CPU_PERF='0.35016'
DVMH_CUDAS_PERF='0.64984'
```

Автоматическое распределение работы между устройствами узла



© ФИЦ ИПМ им. М.В. Келдыша РАН

Выполнение теста EP при одновременном использовании ЦПУ, ГПУ и сопроцессора Xeon PHI



Методы динамической настройки DVMH-программ

- Отображение массивов и циклов на узлы кластера с учетом их производительности
- Отображение массивов и циклов на устройства узла с учетом их производительности
- Трансформация массивов
- Использование динамической компиляции

© ФИЦ ИПМ им. М.В. Келдыша РАН

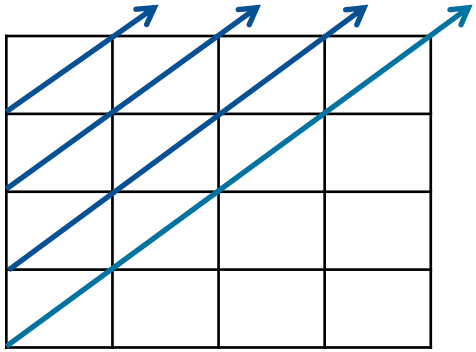
```
#pragma dvm array distribute[block][block]
float A[L][L];
int main(int argc, char *argv[]) {
   float MAXEPS = 0.5E-5f;
   float w = 0.5f;
   for(int it = 0; it < ITMAX; it++) {
     float eps = 0.f;
     #pragma dvm actual(eps)
     #pragma dvm region
      {
         #pragma dvm parallel([i][j] on A[i][j]) across(A[1:1][1:1]), reduction(max(eps))
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L - 1; j++) {
               float s = A[i][j];
               A[i][j] = (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]) / 4.f;
               eps = Max(fabs(s - A[i][j]), eps);
            }
      }
      #pragma dvm get_actual(eps)
      printf("it=%4i   eps=%e\n", it, eps);
      if (eps < MAXEPS) break;
   }
return 0;
}
```

Алгоритм SOR в модели DVMH

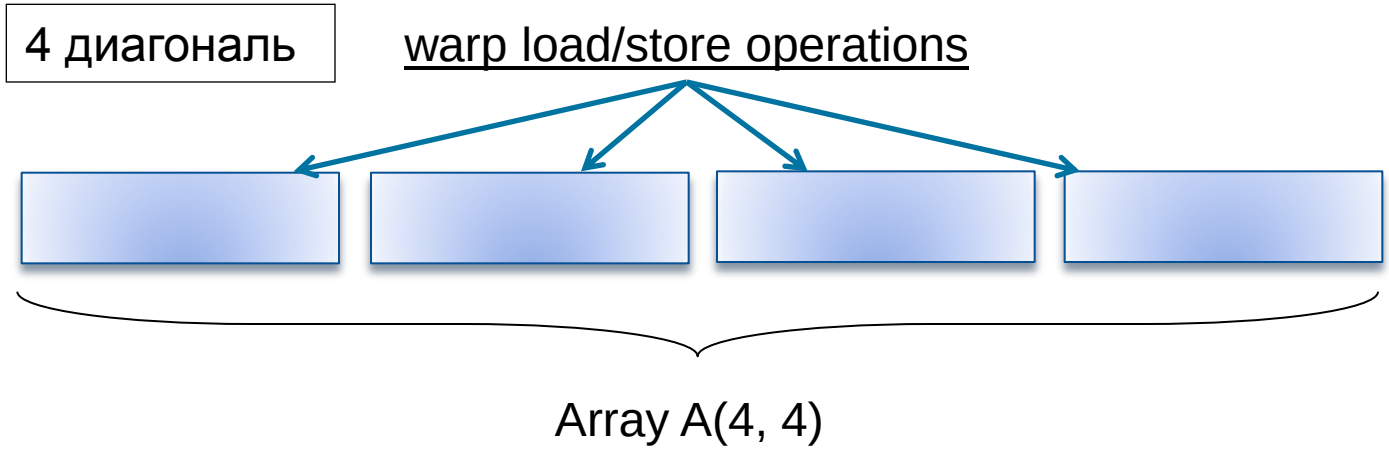
© ФИЦ ИПМ им. М.В. Келдыша РАН

Выполнение гиперплоскостями

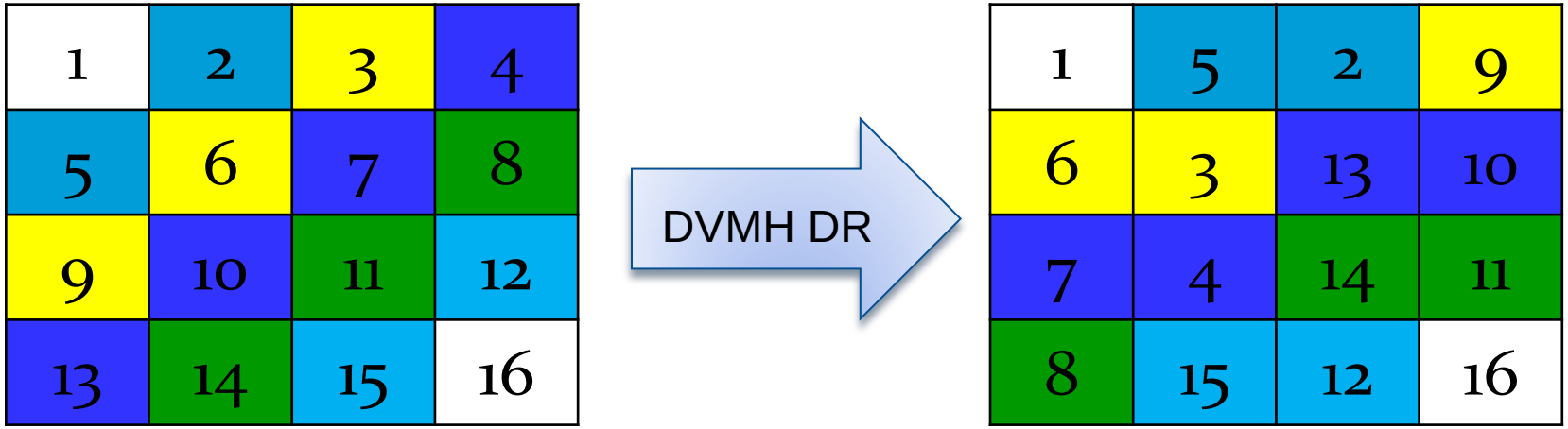
double A[N][N];



```
#pragma dvm parallel([i][j] on A[i][j]), across(A[1:1][1:1])
for (i = 1; i < N - 1; i++)
    for (j = 1; j < N - 1; j++)
        A[i][j] = A[i - 1][j] + A[i + 1][j] + A[i][j - 1] …;
```



Динамическая транформация данных

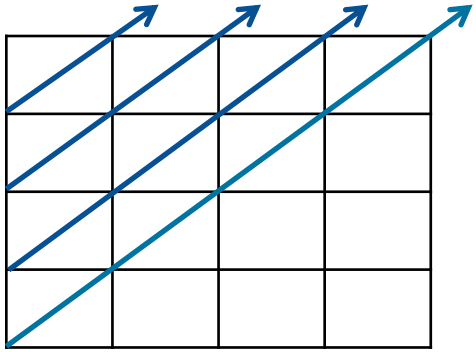


Выполняется подиагональная трансформация матрицы - соседние элементы на диагоналях располагаются в соседних ячейках памяти

© ФИЦ ИПМ им. М.В. Келдыша РАН

Выполнение гиперплоскостями

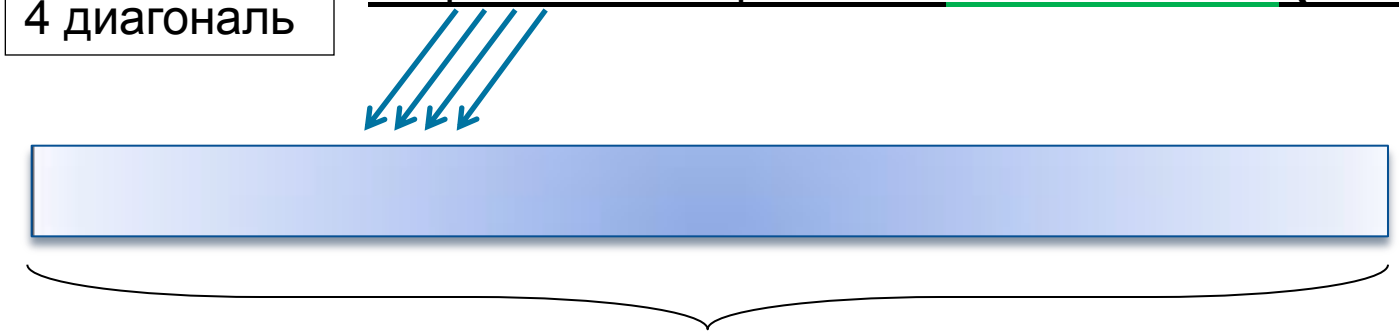
double A[N][N];



```
#pragma dvm parallel([i][j] on A[i][j]), across(A[1:1][1:1])
 for (i = 1; i < N - 1; i++)
      for (j = 1; j < N - 1; j++)
          A[i][j] = A[i - 1][j] + A[i + 1][j] + A[i][j - 1] …;
```

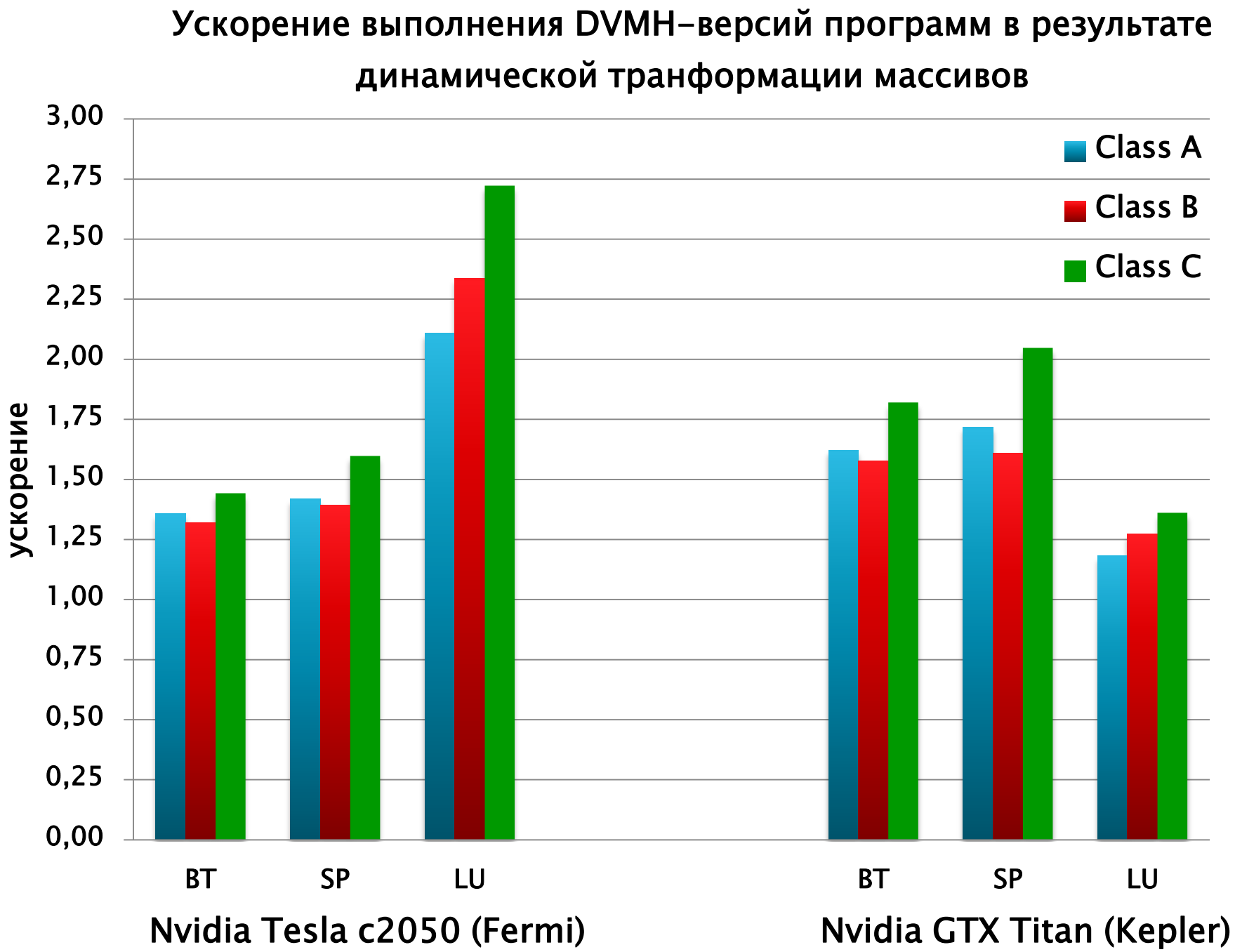
4 диагональ

warp load/store operations **with DVMH DR** **(-autoTfm)**



Array A(4, 4)

© ФИЦ ИПМ им. М.В. Келдыша РАН


Ускорение выполнения DVMH-версий программ в результате динамической транформации массивов
ускорение
3,00
2,75
2,50
2,25
2,00
1,75
1,50
1,25
1,00
0,75
0,50
0,25
0,00
Class A
Class B
Class C
BT
SP
LU
BT
SP
LU
Nvidia Tesla c2050 (Fermi)
Nvidia GTX Titan (Kepler)

Динамическая транформация данных

- Никаких дополнительных указаний в DVMH-программе
- Для каждого цикла для каждого массива выбирается лучший порядок элементов
- Поддержка диагонализированных представлений
- Не происходит возврата состояния в конце цикла, только переход в требуемое

© ФИЦ ИПМ им. М.В. Келдыша РАН

Методы динамической настройки DVMH-программ

- Отображение массивов и циклов на узлы кластера с учетом их производительности
- Отображение массивов и циклов на устройства узла с учетом их производительности
- Трансформация массивов
- Использование динамической компиляции

© ФИЦ ИПМ им. М.В. Келдыша РАН

Динамическая компиляция CUDA-обработчиков во время выполнения программы

- CUDA Toolkit 7.0
- Тест SP из пакета NAS NPB

	Количество регистров			Время выполнения, ms		
	Было	Стало	Сокращение	Было	Стало	Ускорение
compute_rhs	125	82	52%	74	66.5	11%
x_solve	153	112	37%	64.5	42.9	50%
y_solve	132	107	23%	64.4	42.6	51%
z_xolve	128	107	20%	53	48	10%

- Общая производительность всей программы увеличилась на 28%

© ФИЦ ИПМ им. М.В. Келдыша РАН

Новые возможности DVM-системы

- Новая версия компилятора и языка C-DVMH
- Параллельный ввод-вывод
- Новые возможности для функциональной отладки и анализа эффективности параллельных программ

© ФИЦ ИПМ им. М.В. Келдыша РАН

```
FILE *f;
#pragma dvm array distribute[block][block] shadow[1:1][1:1]
double A[L][L];
#pragma dvm array align ([i][j] with A[i][j])
double B[L][L];
int main(int argc, char *argv[]) {
   for(int it = 0; it < ITMAX; it++) {
       #pragma dvm region inout (A,B)
      {
         #pragma dvm parallel([i][j] on A[i][j])
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L-1; j++)  A[i][j] = B[i][j];
         #pragma dvm parallel([i][j] on B[i][j]), shadow_renew(A)
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L - 1; j++)
               B[i][j] = (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]) / 4.;
      }
   }
  f = fopen("jacobi.dat", "wbsl");
  #pragma dvm get_actual(B)
  fwrite(B, sizeof(double), L * L, f);
  fclose(f);
  return 0;
}
```

Алгоритм Якоби в модели DVMH

© ФИЦ ИПМ им. М.В. Келдыша РАН

```
FILE *f;
double A[L][L];
double B[L][L];
int main(int argc, char *argv[]) {
   for(int it = 0; it < ITMAX; it++) {
      #pragma dvm region
      {
         #pragma dvm parallel(2)
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L-1; j++)  A[i][j] = B[i][j];
         #pragma dvm parallel(2)
         for (int i = 1; i < L - 1; i++)
            for (int j = 1; j < L - 1; j++)
               B[i][j] = (A[i - 1][j] + A[i + 1][j] + A[i][j - 1] + A[i][j + 1]) / 4.;
      }
   }
  f = fopen("jacobi.dat", "wbsl");
  #pragma dvm get_actual(B)
  fwrite(B, sizeof(double), L * L, f);
  fclose(f);
  return 0;
}
```

Алгоритм Якоби в модели DVMH

© ФИЦ ИПМ им. М.В. Келдыша РАН

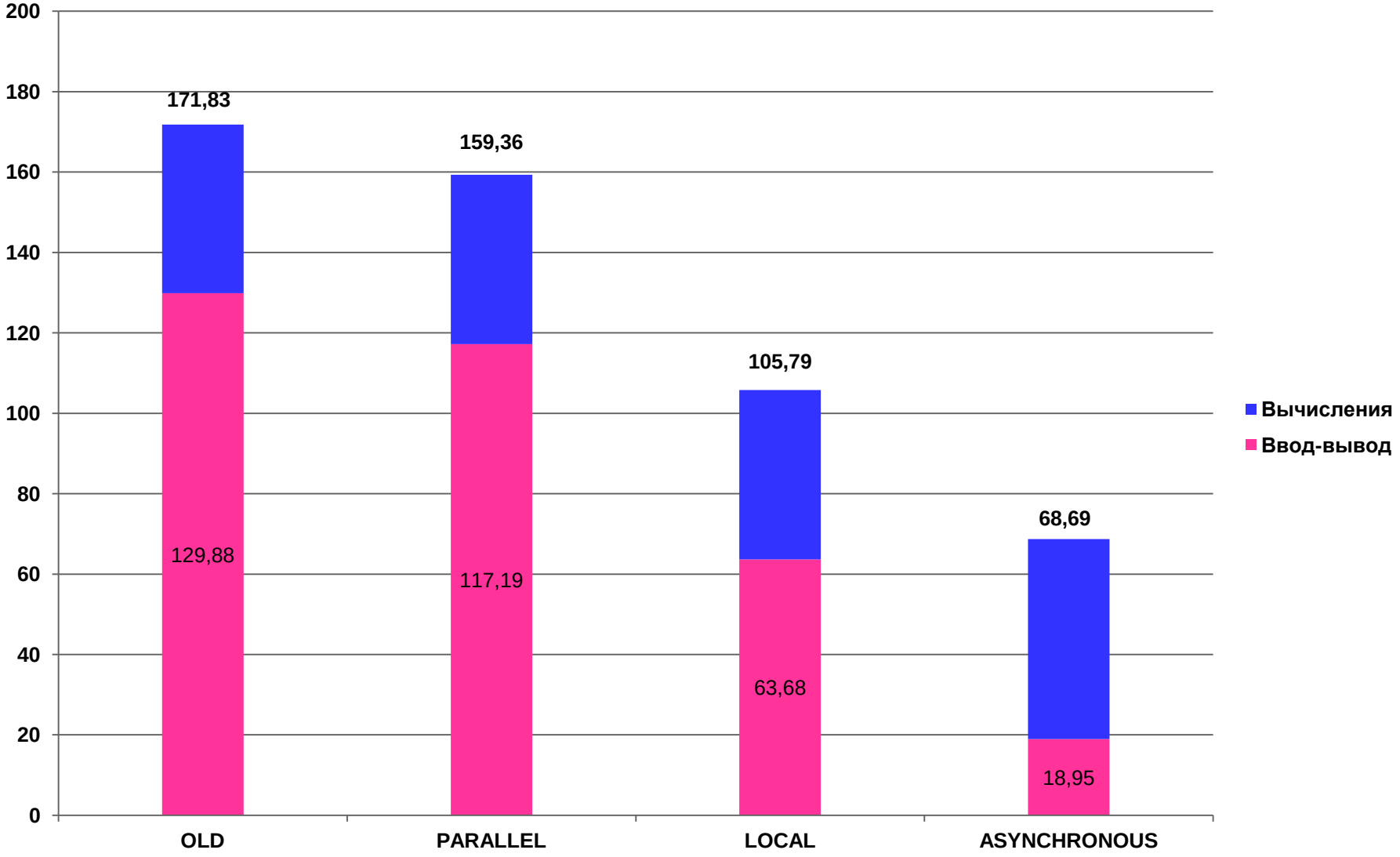
Параллельный ввод-вывод

Реализованы следующие режимы ввода-вывода:

- Все операции осуществляются через один процессор ввода-вывода (OLD)
- Каждый процессор записывает свои данные в свой локальный файл (LOCAL)
- Параллельный ввод-вывод в один глобальный файл (PARALLEL)
- Асинхронный режим, в котором ввод-вывод может выполняться одновременно с вычислениями (ASYNCHRONOUS)

© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения 100 итераций программы Якоби 32000x32000 при сохранении 10 контрольных точек на «Ломоносове»



© ФИЦ ИПМ им. М.В. Келдыша РАН

Сравнительная отладка программы

- Для поиска ошибок используется метод накопления и сравнения результатов вычислений, который позволяет определить место в программе и момент, когда появляются расхождения
- При трассировке вычислений выполняется сбор информации обо всех чтениях и модификациях переменных, о начале выполнения каждого витка цикла
- Степенью подробности и объемом трассировки можно управлять при конвертации программы опциями DVMH-конверторов, а также при выполнении программы
- Существует возможность управлять точностью при сравнении результатов
- Генерируются протоколы с различиями, обнаруженными в процессе сравнения и скачками в значениях переменных и элементах массивов

© ФИЦ ИПМ им. М.В. Келдыша РАН

Новые возможности сравнительной отладки

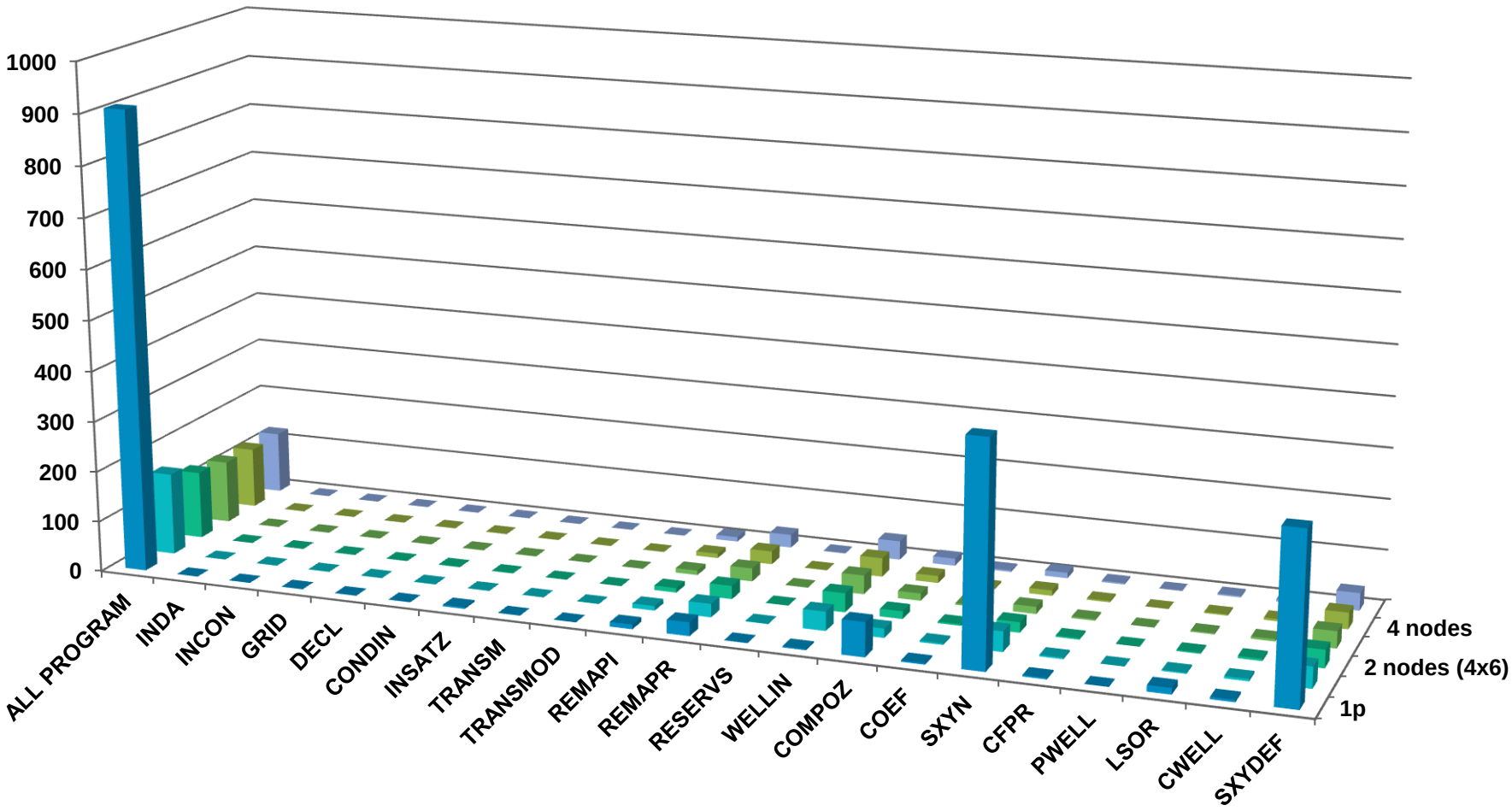
- Специальный режим работы DVMH-программы, при котором все вычисления в регионах одновременно выполняются на ЦПУ и ГПУ
- Сравнение данных осуществляется при входе и выходе из региона
- Позволяет выявить и локализовать ошибки, проявляющиеся при работе на ускорителях
- Целочисленные данные сравниваются на совпадение, а вещественные числа сравниваются с заданной точностью по абсолютной и относительной погрешности
- В случае нахождения расхождений выдается информация о найденных расхождениях
- Далее в программе используется та версия данных, которая была получена при выполнении на ЦПУ

© ФИЦ ИПМ им. М.В. Келдыша РАН

Анализатор эффективности

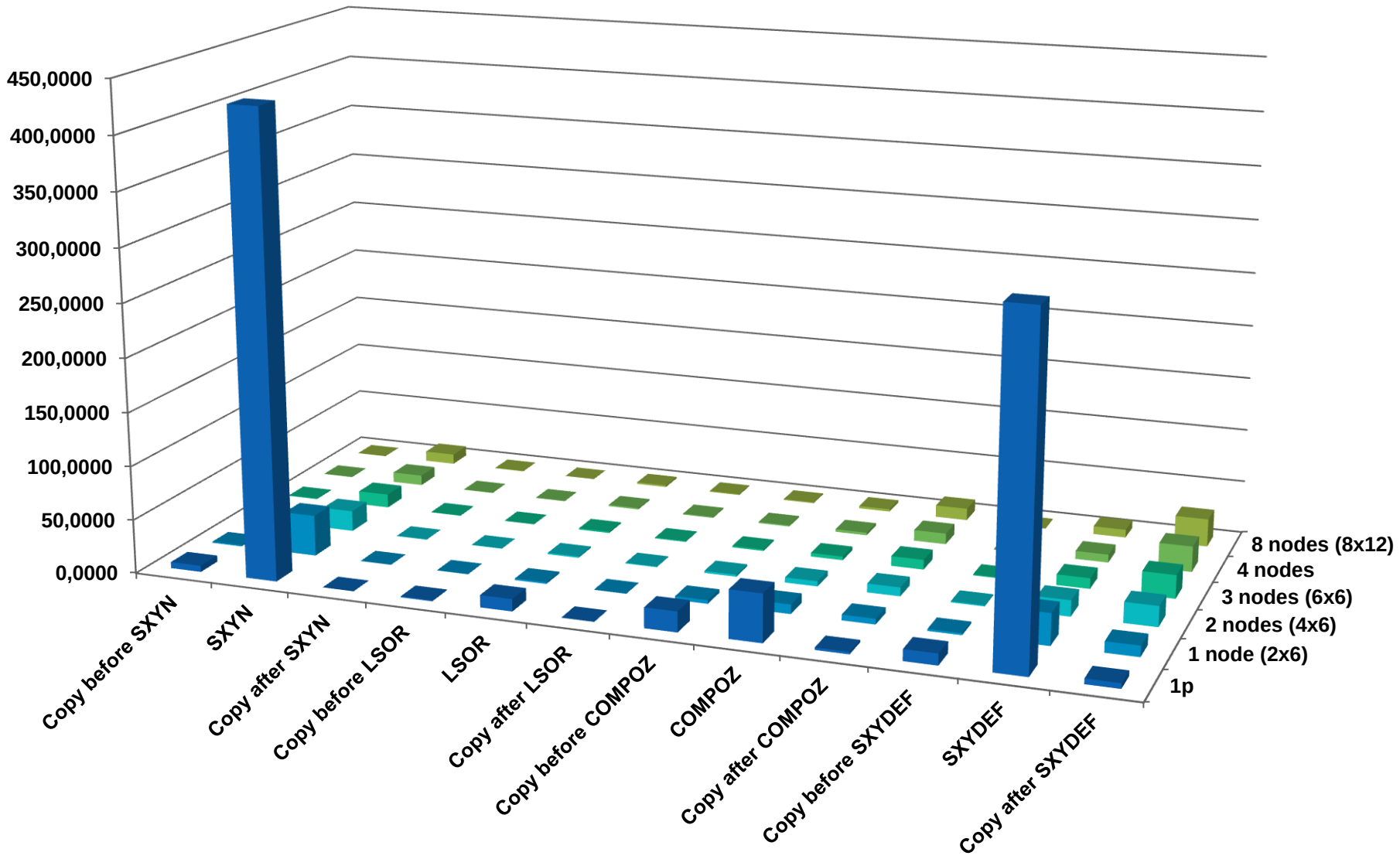
```
INTERVAL ( NLINE=463 SOURCE=COMPOZ.FOR ) LEVEL=3 USER EXE_COUNT=564
-- The main characteristics ---
Parallelization efficiency      1.0000
Execution time                 67.2498
Processors                           1
Total time                     67.2498
Productive time                67.2498 ( CPU= 67.2063  Sys= 0.0435 I/O= 0.0000 )
--- The comparative characteristics ---
                        Tmin N proc      Tmax N proc      Tmid
Execution time          67.2498    1    67.2498    1    67.2498
User CPU time           67.2063    1    67.2063    1    67.2063
Sys. CPU time            0.0435    1     0.0435    1     0.0435
Processors                    1    1         1    1         1
--- The execution characteristics ---
                         1
Execution time          67.2498
User CPU time           67.2063
Sys. CPU time            0.0435
Processors                    1
```

Времена выполнения различных процедур программы Композит



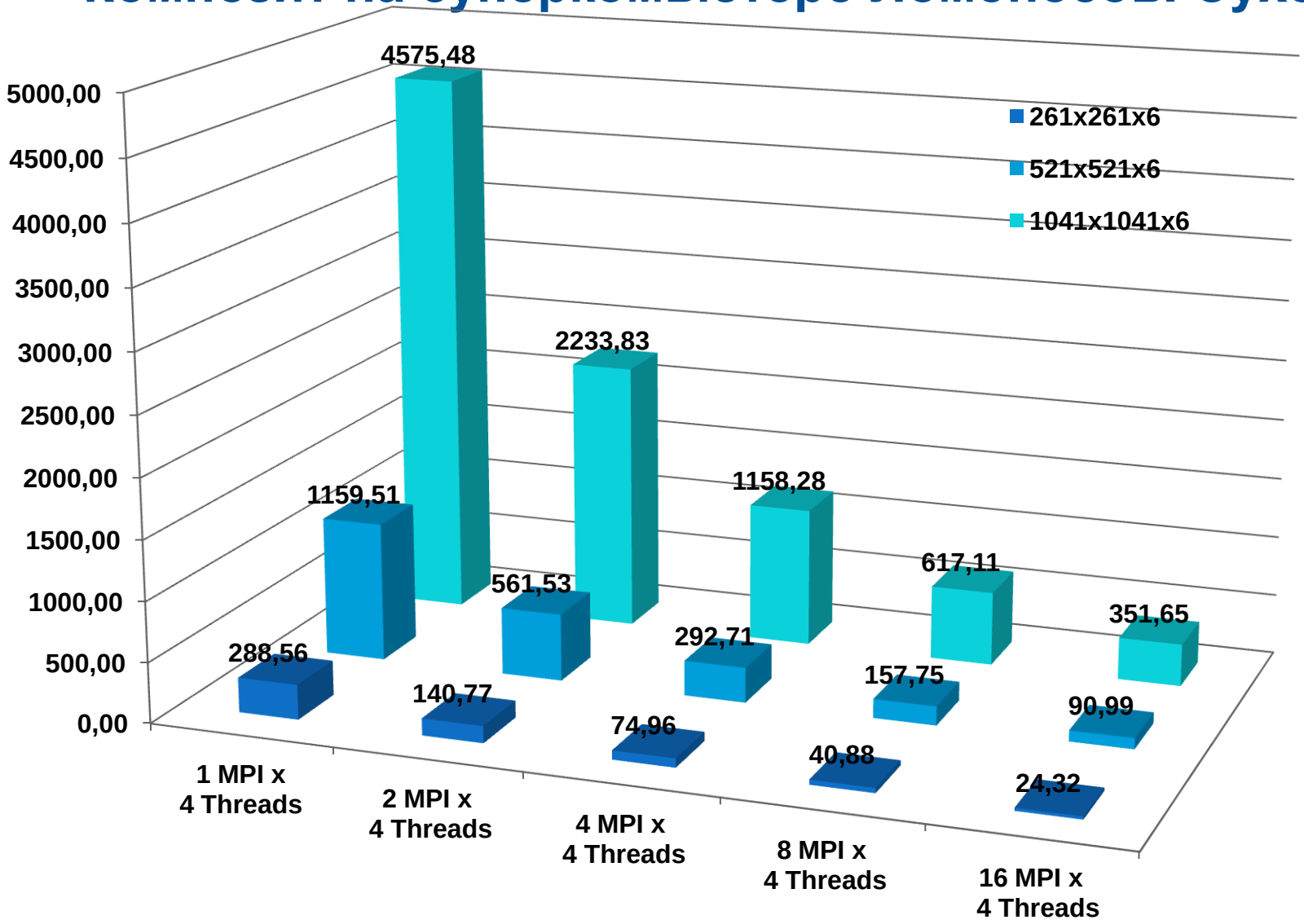
© ФИЦ ИПМ им. М.В. Келдыша РАН

Причины плохого ускорения программы Композит



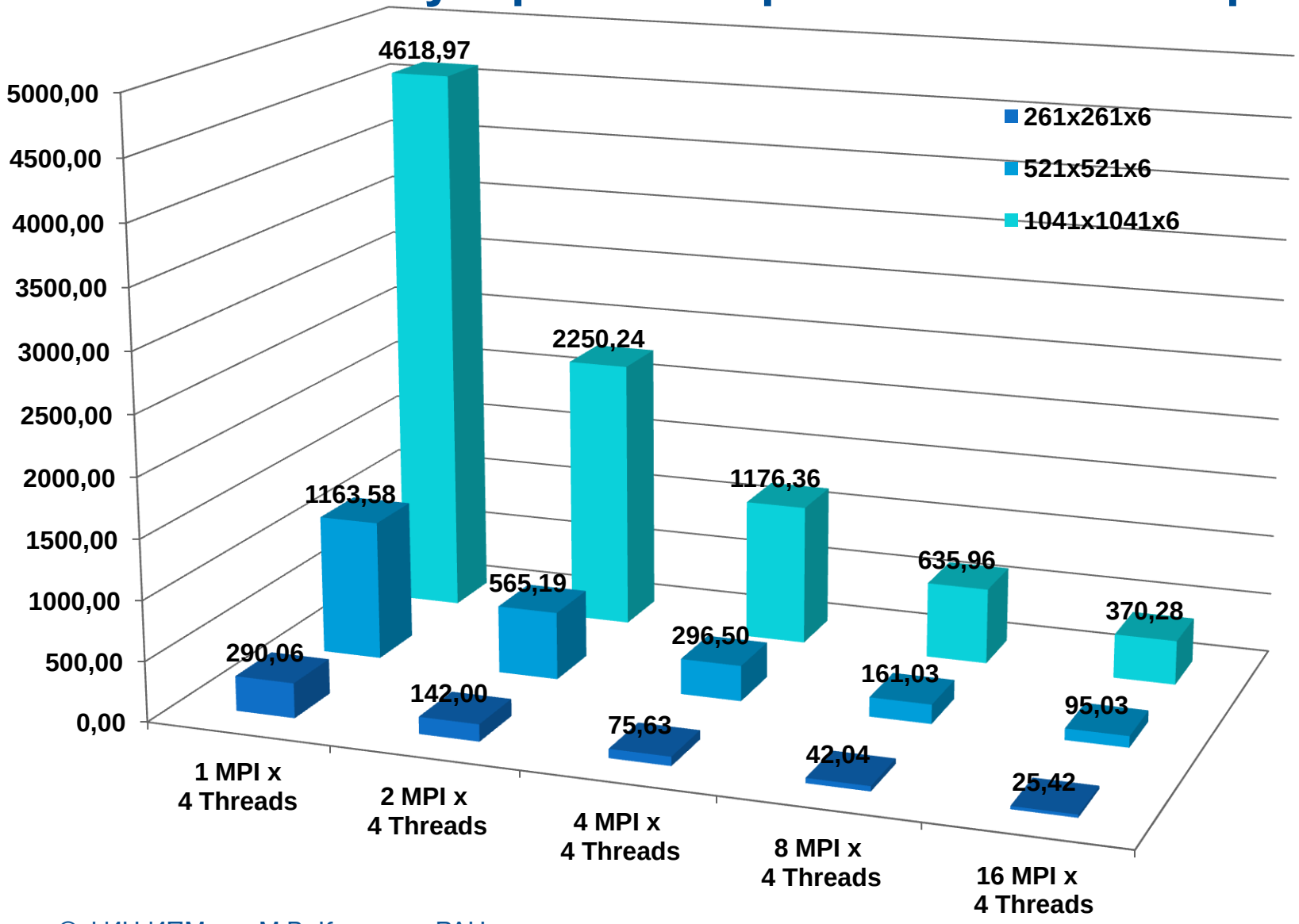
© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения 100 итераций программы Композит на суперкомьютере Ломоносов. Сухой газ



© ФИЦ ИПМ им. М.В. Келдыша РАН

Времена выполнения 100 итераций программы Композит на суперкомьютере Ломоносов. Жирный газ



© ФИЦ ИПМ им. М.В. Келдыша РАН

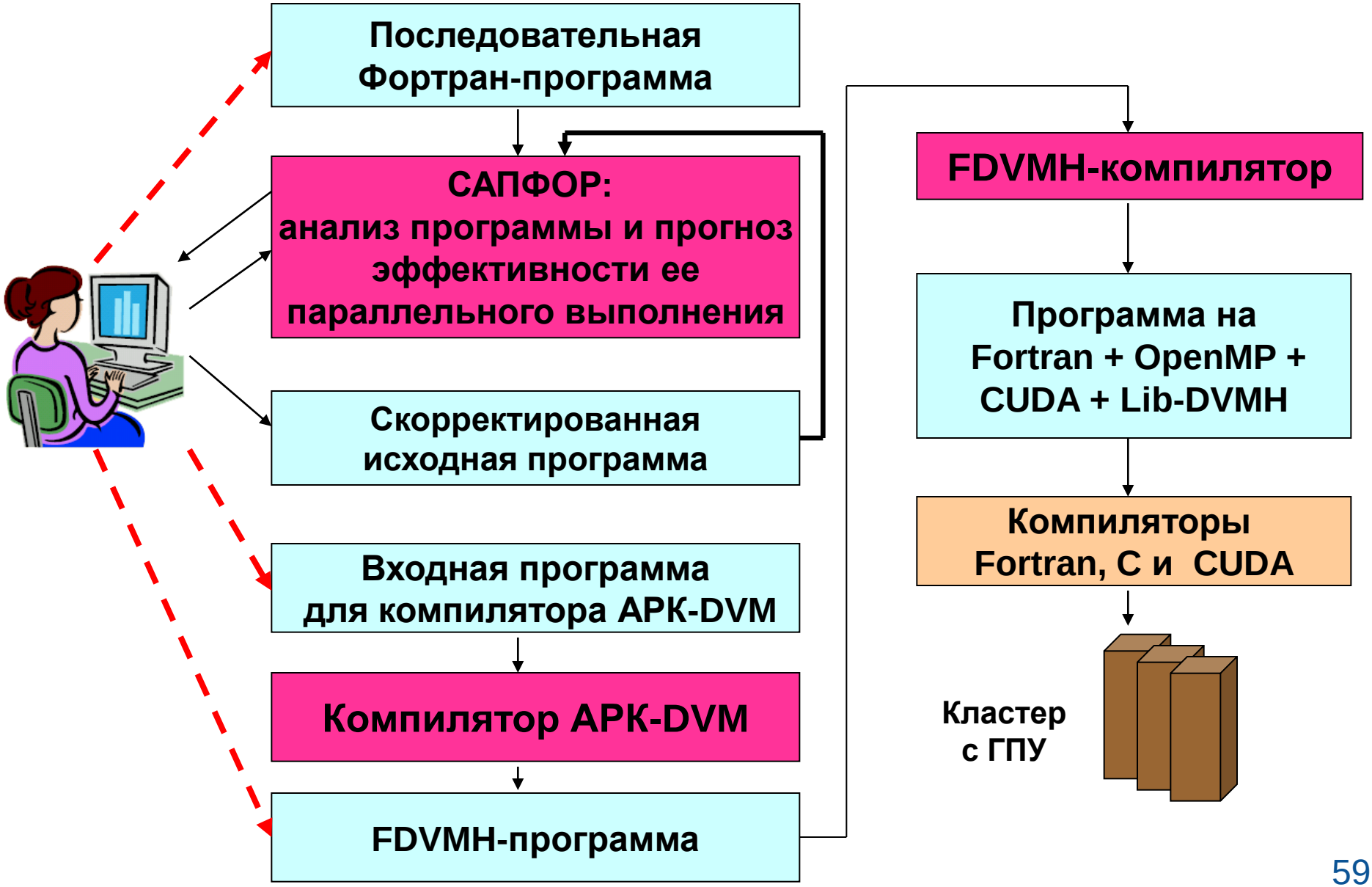
Новые возможности анализатора эффективности

```
--- The GPU characteristics ---

 ┌──────────────┐
 │ Proc: #1     │
 └──────────────┘
       │
   ┌──────────────────────┐
   │ GPU #1 (Tesla X2070) │
   ├──────────────────────┴──────────────────────────────────────────────────────────────────────────────────────┐
   │                                        #          Min          Max          Sum      Average   Productive         Lost │
   ├──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────┤
   │ [Region IN] Copy CPU to GPU            2           0B      976.624M       1.907G      976.624M      0.3252s            - │
   │ GET_ACTUAL                             1           0B      976.624M      976.624M     976.624M      0.1649s            - │
   │ Loop execution time                   41       0.0000        0.0907       2.5133       0.0613      2.5133s            - │
   │ Reduction                             20       0.0000        0.0084       0.0161       0.0008            -      0.0161s │
   ├──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────┤
   │ Productive time:     3.0033s                                                                                          │
   │ Lost time      :     0.0161s                                                                                          │
   └──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────┘

-----------------------------------------------------------------------
Name (number) and performance time of processors
node6-155-07(1) 1.046391
min - node6-155-07(1) 1.046391; max - node6-155-07(1) 1.046391; mid - 1.046391
```

Схема использования DVMH-модели



Система САПОР. Причины отказа в распараллеливании

Visualizer - [cav_1_sep.for]

File Project View Tools Panels Window Help

Files and PU
- cav_1_sep.for
  - cavity

Pro Cal PU

```
do j = 1,ny
 hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
 do i = 0,nx
 t = gm*p(i,j)/ro(i,j)
        tp = gm*p(i+1,j)/ro(i+1,j)
        tm = (t+tp)*0.5d0
        rm = (ro(i,j) + ro(i+1,j))*0.5d0
        pm = (p(i,j) + p(i+1,j))*0.5d0
        uxm = (ux(i,j) + ux(i+1,j))*0.5d0
```

Loo...	Progr...	File name	dvm_xp_0 Decisions	Writ
=,,	cavity	cav_1_sep.for (58)	наличие операторов ввода-вывода	
i=1,nx	cavity	cav_1_sep.for (71)	данные для записи витка ro(i,0) и ro(i,ny1) находятся на разных процессорах	
j=1,ny	cavity	cav_1_sep.for (85)	данные для записи витка ro(0,j) и ro(nx1,j) находятся на разных процессорах	
j=1,ny	cavity	cav_1_sep.for (147)	данные для записи витка tmp1(:,j) находятся на разных процессорах	
i=1,nx	cavity	cav_1_sep.for (206)	данные для записи витка tmp1(i,:) находятся на разных процессорах	
j=1,ny	cavity	cav_1_sep.for (291)	данные для записи витка tmp1(:,j) находятся на разных процессорах	
i=1,nx	cavity	cav_1_sep.for (344)	данные для записи витка tmp1(i,:) находятся на разных процессорах	
i=1,nx	cavity	cav_1_sep.for (454)	данные для записи витка ro(i,0) и ro(i,ny1) находятся на разных процессорах	
j=1,ny	cavity	cav_1_sep.for (468)	данные для записи витка ro(0,j) и ro(nx1,j) находятся на разных процессорах	
j=1,ny1	cavity	cav_1_sep.for (527)	наличие операторов ввода-вывода	
i=1,nx1	cavity	cav_1_sep.for (529)	наличие операторов ввода-вывода	

Output | Decisions | Variables | Characteristics

Ready | Ln 147

Система САПОР. Прогнозирование параллельного выполнения

Visualizer - [cav_1_sep.for]

File Project View Tools Panels Window Help

Files and PU
- cav_1_sep.for
  - cavity

Pro Cal PU

```
do j = 1,ny
 hy4 = hy(j+1)+2.d0*hy(j)+hy(j-1)
 do i = 0,nx
 t = gm*p(i,j)/ro(i,j)
        tp = gm*p(i+1,j)/ro(i+1,j)
        tm = (t+tp)*0.5d0
        rm = (ro(i,j) + ro(i+1,j))*0.5d0
        pm = (p(i,j) + p(i+1,j))*0.5d0
        uxm = (ux(i,j) + ux(i+1,j))*0.5d0
```

Loops ...	Program ...	File name	Level ...	Sequential time ...	Parallel time ...	Speed up...	Communication...	Parallel ...
=,,	cavity	cav_1_sep.for (58)	1.4	600.643616	609.380737	0.985662	2967.039551	no
j=1,ny	cavity	cav_1_sep.for (147)	1.4.5	116.019997	195.772003	0.592628	1065.375977	no
i=0,nx	cavity	cav_1_sep.for (149)	1.4.5.1	116.000000	195.751999	0.592587	1065.375977	yes
i=1,nx	cavity	cav_1_sep.for (130)	1.4.4	20.000000	187.844864	0.106471	755.202515	yes
j=1,ny	cavity	cav_1_sep.for (291)	1.4.11	108.020004	151.647995	0.712307	514.943970	no
i=0,nx	cavity	cav_1_sep.for (293)	1.4.11.1	108.000008	151.628006	0.712270	514.943970	yes
i=0,nx1	cavity	cav_1_sep.for (60)	1.4.1	16.160400	102.586884	0.157529	432.970886	yes
i=1,nx	cavity	cav_1_sep.for (206)	1.4.7	116.019997	61.447998	1.888101	184.384003	no
j=0,ny	cavity	cav_1_sep.for (208)	1.4.7.1	116.000000	61.427998	1.888390	184.384003	yes
i=1,nx	cavity	cav_1_sep.for (344)	1.4.13	108.020004	31.560001	3.422687	0.000000	no
j=0,ny	cavity	cav_1_sep.for (346)	1.4.13.1	108.000008	31.540001	3.424223	0.000000	yes
i=1,nx	cavity	cav_1_sep.for (410)	1.4.16	44.000000	6.790580	6.479564	9.687840	yes
i=1,nx	cavity	cav_1_sep.for (275)	1.4.10	24.000000	3.110600	7.715553	0.000000	yes

Output | Decisions | Variables | Characteristics

Ready | Ln 149

Выводы

- DVM-система существенно упрощает процесс разработки параллельных программ для гибридных вычислительных кластеров.
- Получаемые DVMH-программы без каких-либо изменений могут эффективно выполняться на кластерах различной архитектуры, использующих многоядерные универсальные процессоры, графические ускорители и сопроцессоры Intel Xeon Phi. Это достигается как за счет различных оптимизаций, которые выполняются статически, при компиляции DVMH-программ, так и за счет динамических оптимизаций.
- Параллельные программы могут динамически настраиваться при запуске на выделенные для их выполнения ресурсы (количество узлов кластера, ядер, ускорителей и их производительность).

© ФИЦ ИПМ им. М.В. Келдыша РАН

Вопросы, замечания?

СПАСИБО !

http://dvm-system.org

dvm@keldysh.ru