



Спецсеминар

«Автоматизация параллельного программирования»

Бахтин Владимир Александрович

Доцент кафедры системного программирования

факультета ВМК, МГУ им. М. В. Ломоносова

к.ф.-м.н., вед. научный сотрудник Института прикладной

математики им М.В.Келдыша РАН

bakhtin@keldysh.ru

МГУ им. М.В. Ломоносова, Москва, 2022 г.



Содержание

- ❑ Современные направления развития параллельных вычислительных систем
- ❑ Технологии параллельного программирования
- ❑ Система SAPFOR (System FOR Automated Parallelization)



Крюков Виктор Алексеевич
Профессор кафедры системного
программирования факультета ВМК
МГУ им. М.В. Ломоносова
д.ф.-м.н., главный научный сотрудник
Института прикладной математики
им М.В.Келдыша РАН
krukov@keldysh.ru



PRESENTED BY



FIND OUT MORE AT

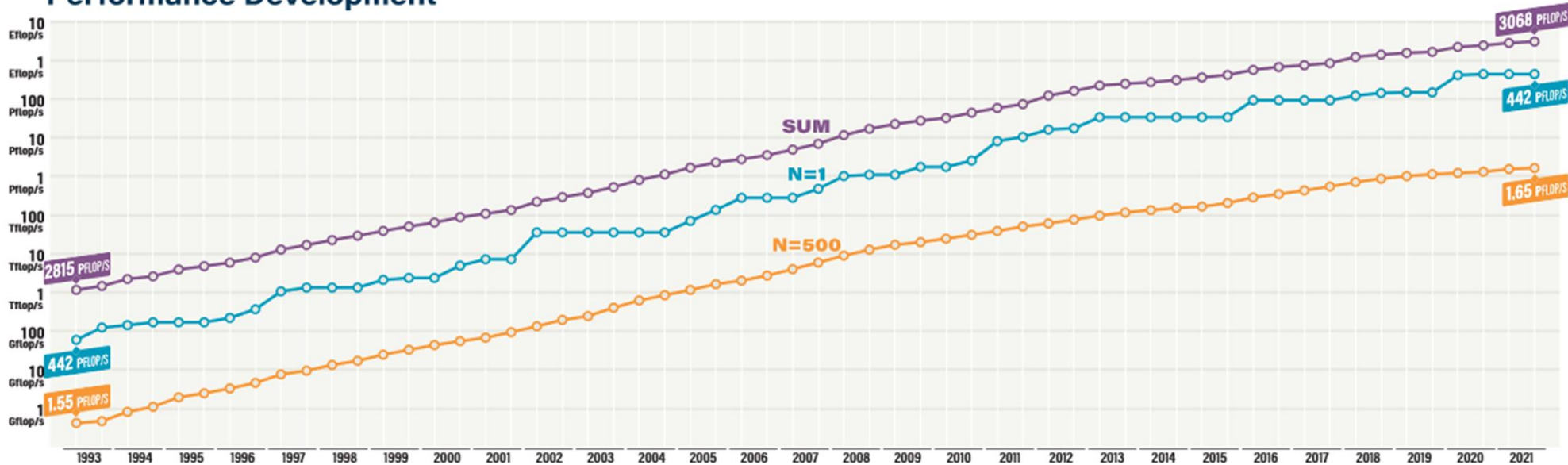
top500.org



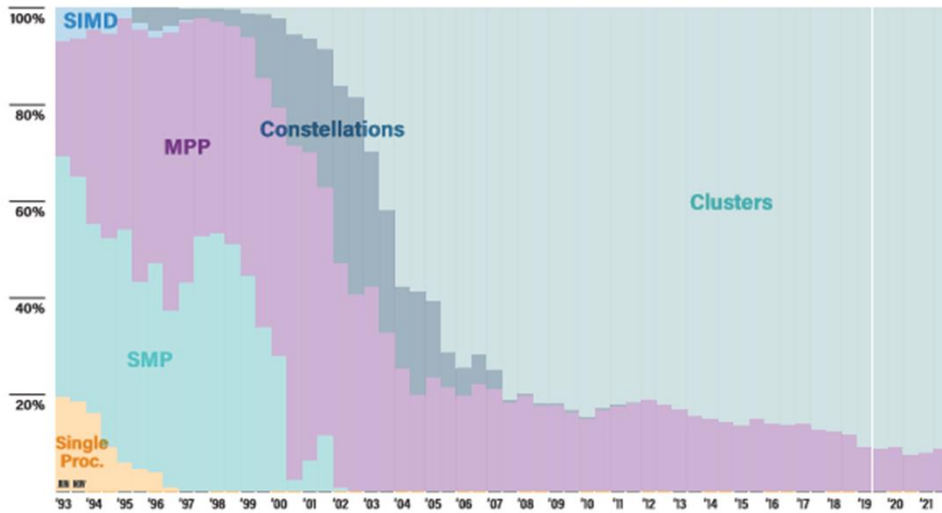
NOVEMBER 2021

			SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
2	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
3	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44
4	Sunway TaihuLight	Shenwei SW26010 (260C, 1.45 GHz) Custom Interconnect	NSSC in Wuxi	China	10,649,600	93.0	15.4
5	Perlmutter	HPE Cray EX235n, AMD EPYC 7763 64C 2.45GHz, NVIDIA A100 SXM4 40 GB, Slingshot-10 (274 GB)	LBNL	USA	761,856	70.9	2.58

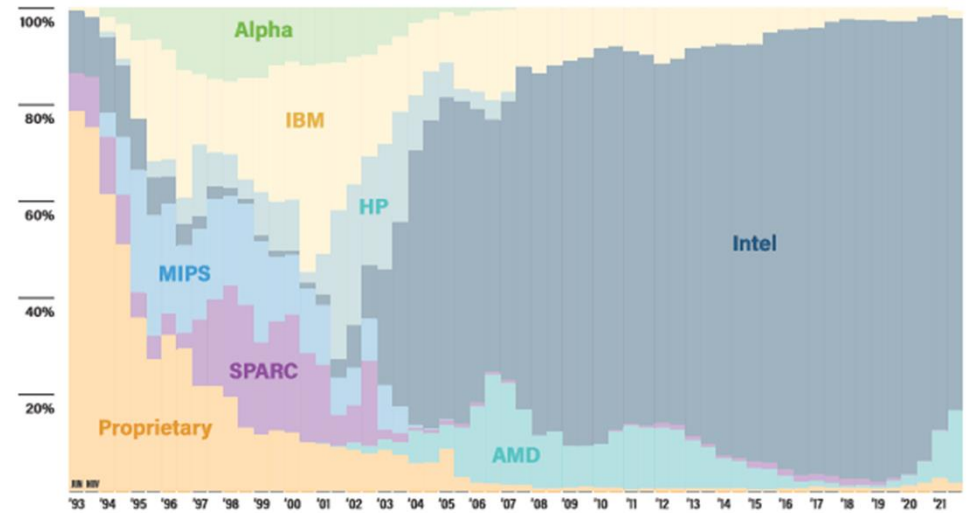
Performance Development



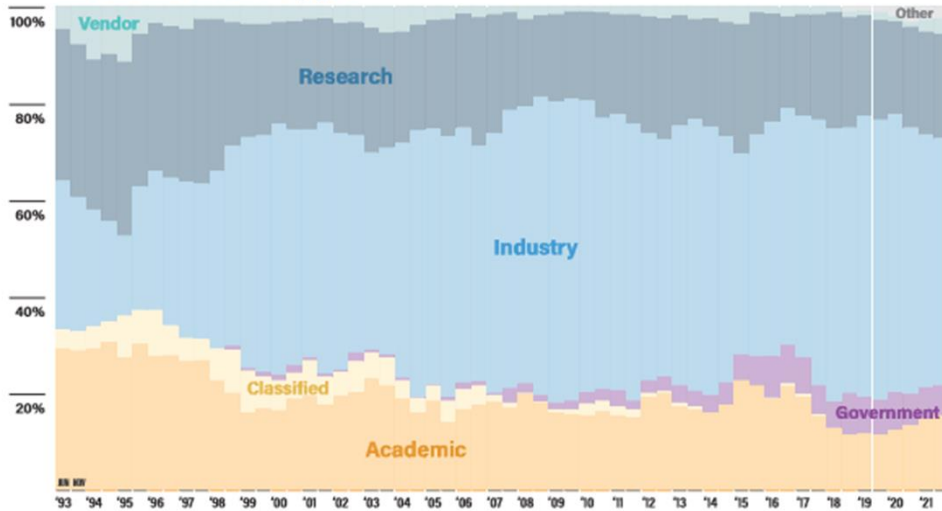
Architectures



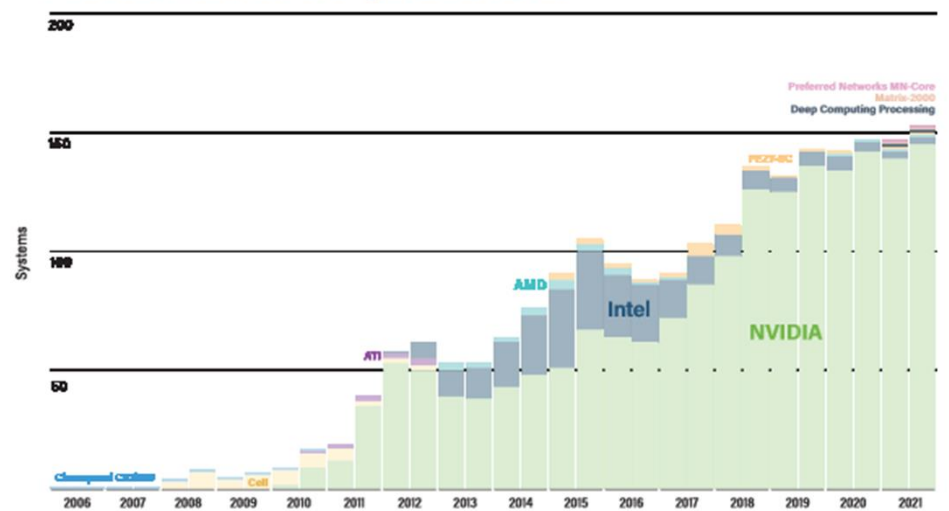
Chip Technology



Installation Type



Accelerators/Co-processors





Содержание

- ❑ Современные направления развития параллельных вычислительных систем
- ❑ Технологии параллельного программирования
- ❑ Система SAPFOR (System FOR Automated Parallelization)



Алгоритм Якоби. Последовательная версия

```
/* Jacobi program */
#include <stdio.h>
#define L 1000
#define ITMAX 100
int i,j,it;
double A[L][L];
double B[L][L];
int main(int an, char **as)
{
    printf("JAC STARTED\n");
    for(i=0;i<=L-1;i++)
        for(j=0;j<=L-1;j++)
        {
            A[i][j]=0.;
            B[i][j]=1.+i+j;
        }
}
```

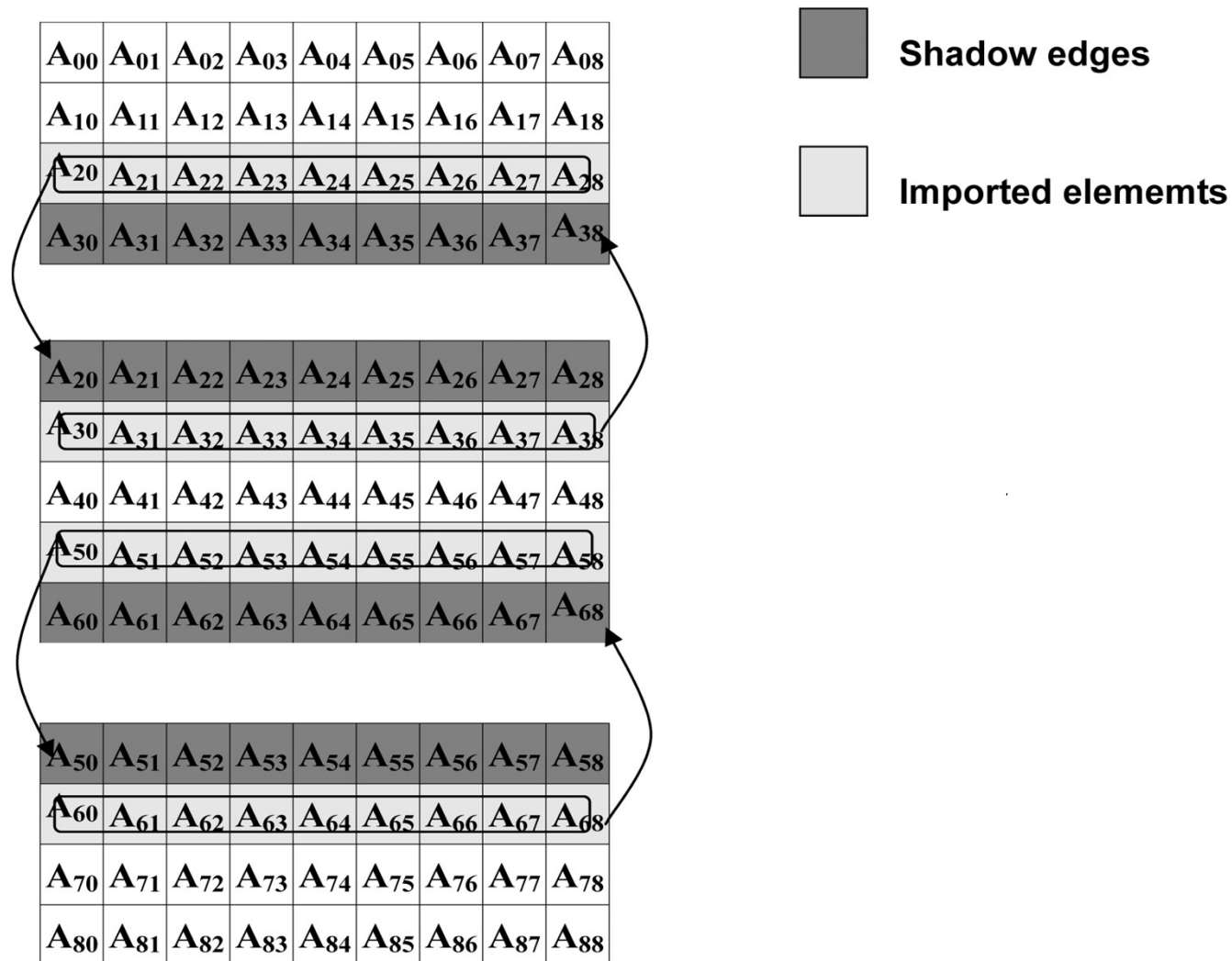


Алгоритм Якоби. Последовательная версия

```
/****** iteration loop *****/
for(it=1; it<ITMAX;it++)
{
    for(i=1;i<=L-2;i++)
        for(j=1;j<=L-2;j++)
            A[i][j] = B[i][j];
    for(i=1;i<=L-2;i++)
        for(j=1;j<=L-2;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. MPI-версия





Алгоритм Якоби. MPI-версия

```
/* Jacobi-1d program */
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include "mpi.h"
#define m_printf if (myrank==0)printf
#define L 1000
#define ITMAX 100

int i,j,it,k;
int ll,shift;
double (* A)[L];
double (* B)[L];
```



Алгоритм Якоби. MPI-версия

```
int main(int argc, char **argv)
{
    MPI_Request req[4];
    int myrank, ranksize;
    int startrow, lastrow, nrow;
    MPI_Status status[4];
    double t1, t2, time;
    MPI_Init (&argc, &argv); /* initialize MPI system */
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank); /* my place in MPI system */
    MPI_Comm_size (MPI_COMM_WORLD, &ranksize); /* size of MPI system */
    MPI_Barrier(MPI_COMM_WORLD);
    /* rows of matrix I have to process */
    startrow = (myrank * L) / ranksize;
    lastrow = (((myrank + 1) * L) / ranksize) - 1;
    nrow = lastrow - startrow + 1;
    m_printf("JAC1 STARTED\n");
}
```



Алгоритм Якоби. MPI-версия

```
/* dynamically allocate data structures */
A = malloc ((nrow+2) * L * sizeof(double));
B = malloc ((nrow) * L * sizeof(double));
for(i=1; i<=nrow; i++)
    for(j=0; j<=L-1; j++)
    {
        A[i][j]=0.;
        B[i-1][j]=1.+startrow+i-1+j;
    }
```



Алгоритм Якоби. MPI-версия

```
/****** iteration loop *****/
t1=MPI_Wtime();
for(it=1; it<=ITMAX; it++)
{
    for(i=1; i<=nrow; i++)
    {
        if (((i==1)&&(myrank==0))||((i==nrow)&&(myrank==ranksize-1)))
            continue;
        for(j=1; j<=L-2; j++)
        {
            A[i][j] = B[i-1][j];
        }
    }
}
```



Алгоритм Якоби. MPI-версия

```
if(myrank!=0)
    MPI_Irecv(&A[0][0],L,MPI_DOUBLE, myrank-1, 1215,
             MPI_COMM_WORLD, &req[0]);
if(myrank!=ranksize-1)
    MPI_Isend(&A[nrow][0],L,MPI_DOUBLE, myrank+1, 1215,
             MPI_COMM_WORLD,&req[2]);
if(myrank!=ranksize-1)
    MPI_Irecv(&A[nrow+1][0],L,MPI_DOUBLE, myrank+1, 1216,
             MPI_COMM_WORLD, &req[3]);
if(myrank!=0)
    MPI_Isend(&A[1][0],L,MPI_DOUBLE, myrank-1, 1216,
             MPI_COMM_WORLD,&req[1]);
ll=4; shift=0;
if (myrank==0) {ll=2;shift=2;}
if (myrank==ranksize-1) {ll=2;}
MPI_Waitall(ll,&req[shift],&status[0]);
```



Алгоритм Якоби. MPI-версия

```
for(i=1; i<=nrow; i++)
{
    if (((i==1)&&(myrank==0))||((i==nrow)&&(myrank==ranksize-1))) continue;
    for(j=1; j<=L-2; j++)
        B[i-1][j] = (A[i-1][j]+A[i+1][j]+
                    A[i][j-1]+A[i][j+1])/4.;
}
}/*DO it*/
printf("%d: Time of task=%f\n",myrank,MPI_Wtime()-t1);
MPI_Finalize ();
return 0;
}
```




Алгоритм Якоби. MPI-версия

```
/*Jacobi-2d program */
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include "mpi.h"
#define m_printf if (myrank==0)printf
#define L 1000
#define LC 2
#define ITMAX 100

int i,j,it,k;

double (* A)[L/LC+2];
double (* B)[L/LC];
```



Алгоритм Якоби. MPI-версия

```
int main(int argc, char **argv)
{
MPI_Request req[8];
int myrank, ranksize;
int srow,lrow,nrow,scol,lcol,ncol;
MPI_Status status[8];
double t1;
int isper[] = {0,0};
int dim[2];
int coords[2];
MPI_Comm newcomm;
MPI_Datatype vectype;
int pleft,pright, pdown,pup;
MPI_Init (&argc, &argv);      /* initialize MPI system */
MPI_Comm_size (MPI_COMM_WORLD, &ranksize); /* size of MPI system */
MPI_Comm_rank (MPI_COMM_WORLD, &myrank);  /* my place in MPI system */
```



Алгоритм Якоби. MPI-версия

```
dim[0]=ranksize/LC;
dim[1]=LC;
if ((L%dim[0])||(L%dim[1]))
{
    m_printf("ERROR: array[%d*%d] is not distributed on %d*%d
processors\n",L,L,dim[0],dim[1]);
    MPI_Finalize();
    exit(1);
}
MPI_Cart_create(MPI_COMM_WORLD,2,dim,isper,1,&newcomm);
MPI_Cart_shift(newcomm,0,1,&pup,&pdown);
MPI_Cart_shift(newcomm,1,1,&pleft,&pright);
MPI_Comm_rank (newcomm, &myrank); /* my place in MPI system */
MPI_Cart_coords(newcomm,myrank,2,coords);
```



Алгоритм Якоби. MPI-версия

```
/* rows of matrix I have to process */  
srow = (coords[0] * L) / dim[0];  
lrow = (((coords[0] + 1) * L) / dim[0])-1;  
nrow = lrow - srow + 1;  
/* columns of matrix I have to process */  
scol = (coords[1] * L) / dim[1];  
lcol = (((coords[1] + 1) * L) / dim[1])-1;  
ncol = lcol - scol + 1;  
MPI_Type_vector(nrow,1,ncol+2,MPI_DOUBLE,&vectype);  
MPI_Type_commit(&vectype);  
m_printf("JAC2 STARTED on %d*%d processors with %d*%d array,  
it=%d\n",dim[0],dim[1],L,L,ITMAX);  
/* dynamically allocate data structures */  
A = malloc ((nrow+2) * (ncol+2) * sizeof(double));  
B = malloc (nrow * ncol * sizeof(double));
```



Алгоритм Якоби. MPI-версия

```
for(i=0; i<=nrow-1; i++)
{
    for(j=0; j<=ncol-1; j++)
    {
        A[i+1][j+1]=0.;
        B[i][j]=1.+srow+i+scol+j;
    }
}
/***** iteration loop *****/
MPI_Barrier(newcomm);
t1=MPI_Wtime();
for(it=1; it<=ITMAX; it++)
{
    for(i=0; i<=nrow-1; i++)
    {
        if (((i==0)&&(pup==MPI_PROC_NULL))||((i==nrow-1)&&(pdown==MPI_PROC_NULL))) continue;
        for(j=0; j<=ncol-1; j++)
        {
            if (((j==0)&&(pleft==MPI_PROC_NULL))||((j==ncol-1)&&(pright==MPI_PROC_NULL)))
                continue;
            A[i+1][j+1] = B[i][j];
        }
    }
}
```



Алгоритм Якоби. MPI-версия

```
MPI_Irecv(&A[0][1],ncol,MPI_DOUBLE,
  pup, 1215, MPI_COMM_WORLD, &req[0]);
MPI_Isend(&A[nrow][1],ncol,MPI_DOUBLE,
  pdown, 1215, MPI_COMM_WORLD,&req[1]);
MPI_Irecv(&A[nrow+1][1],ncol,MPI_DOUBLE,
  pdown, 1216, MPI_COMM_WORLD, &req[2]);
MPI_Isend(&A[1][1],ncol,MPI_DOUBLE,
  pup, 1216, MPI_COMM_WORLD,&req[3]);
MPI_Irecv(&A[1][0],1,vectype,
  pleft, 1217, MPI_COMM_WORLD, &req[4]);
MPI_Isend(&A[1][ncol],1,vectype,
  pright, 1217, MPI_COMM_WORLD,&req[5]);
MPI_Irecv(&A[1][ncol+1],1,vectype,
  pright, 1218, MPI_COMM_WORLD, &req[6]);
MPI_Isend(&A[1][1],1,vectype,
  pleft, 1218, MPI_COMM_WORLD,&req[7]);
MPI_Waitall(8,req,status);
```



Алгоритм Якоби. MPI-версия

```
for(i=1; i<=nrow; i++)
{
    if (((i==1)&&(pup==MPI_PROC_NULL))||
        ((i==nrow)&&(pdown==MPI_PROC_NULL))) continue;
    for(j=1; j<=ncol; j++)
    {
        if (((j==1)&&(pleft==MPI_PROC_NULL))||
            ((j==ncol)&&(pright==MPI_PROC_NULL))) continue;
        B[i-1][j-1] = (A[i-1][j]+A[i+1][j]+A[i][j-1]+A[i][j+1])/4.;
    }
}
printf("%d: Time of task=%lf\n",myrank,MPI_Wtime()-t1);
MPI_Finalize ();
return 0;
}
```



Алгоритм Якоби. OpenMP-версия

```
/* Jacobi program */
#include <stdio.h>
#define L 1000
#define ITMAX 100
int i,j,it;
double A[L][L];
double B[L][L];
int main(int an, char **as)
{
    printf("JAC STARTED\n");
    #pragma omp parallel for private(i,j)
    for(i=0;i<=L-1;i++)
        for(j=0;j<=L-1;j++)
        {
            A[i][j]=0.;
            B[i][j]=1.+i+j;
        }
}
```

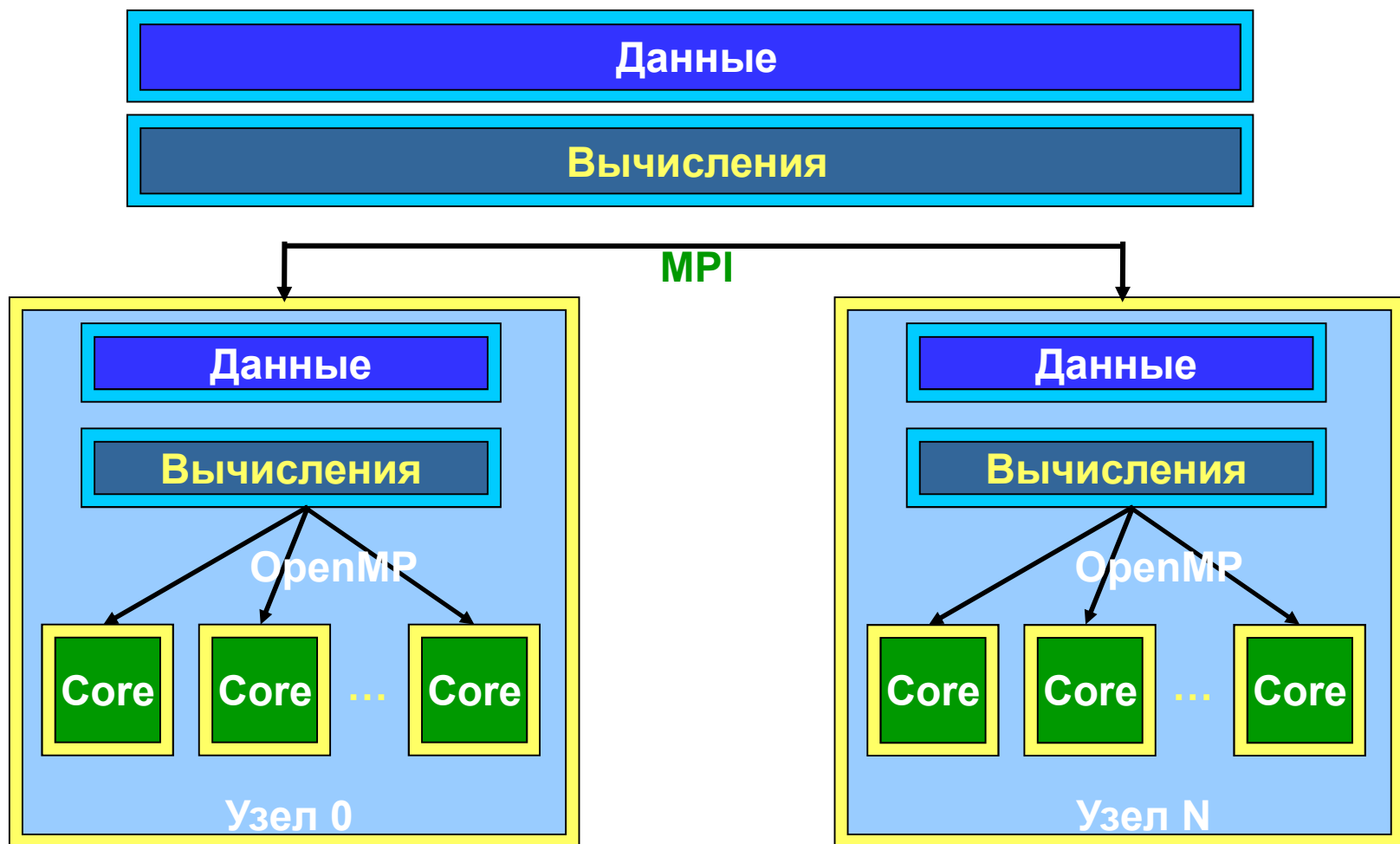


Алгоритм Якоби. OpenMP-версия

```
/****** iteration loop *****/
for(it=1; it<ITMAX;it++)
{
    #pragma omp parallel for private(i,j)
    for(i=1;i<=L-2;i++)
        for(j=1;j<=L-2;j++)
            A[i][j] = B[i][j];
    #pragma omp parallel for private(i,j)
    for(i=1;i<=L-2;i++)
        for(j=1;j<=L-2;j++)
            B[i][j] = (A[i-1][j]+A[i+1][j]+A[i][j-1]+A[i][j+1])/4.;
}
return 0;
}
```



Гибридная модель MPI/OpenMP





Алгоритм Якоби. MPI/OpenMP-версия

```
/* Jacobi-1d program */
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include "mpi.h"
#define m_printf if (myrank==0)printf
#define L 1000
#define ITMAX 100

int i,j,it,k;
int ll,shift;
double (* A)[L];
double (* B)[L];
```



Алгоритм Якоби. MPI/OpenMP-версия

```
int main(int argc, char **argv)
{
    MPI_Request req[4];
    int myrank, ranksize;
    int startrow, lastrow, nrow;
    MPI_Status status[4];
    double t1, t2, time;
    MPI_Init (&argc, &argv); /* initialize MPI system */
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank); /*my place in MPI system */
    MPI_Comm_size (MPI_COMM_WORLD, &ranksize); /* size of MPI system */
    MPI_Barrier(MPI_COMM_WORLD);
    /* rows of matrix I have to process */
    startrow = (myrank * N) / ranksize;
    lastrow = (((myrank + 1) * N) / ranksize) - 1;
    nrow = lastrow - startrow + 1;
    m_printf("JAC1 STARTED\n");
}
```



Алгоритм Якоби. MPI/OpenMP-версия

```
/* dynamically allocate data structures */  
A = malloc ((nrow+2) * N * sizeof(double));  
B = malloc ((nrow) * N * sizeof(double));  
for(i=1; i<=nrow; i++)  
    #pragma omp parallel for  
    for(j=0; j<=L-1; j++)  
    {  
        A[i][j]=0.;  
        B[i-1][j]=1.+startrow+i-1+j;  
    }
```



Алгоритм Якоби. MPI/OpenMP-версия

```
/****** iteration loop *****/
t1=MPI_Wtime();
for(it=1; it<=ITMAX; it++)
{
    for(i=1; i<=nrow; i++)
    {
        if (((i==1)&&(myrank==0))||((i==nrow)&&(myrank==ranksize-1)))
            continue;
        #pragma omp parallel for
        for(j=1; j<=L-2; j++)
        {
            A[i][j] = B[i-1][j];
        }
    }
}
```



Алгоритм Якоби. MPI/OpenMP-версия

```
if(myrank!=0)
    MPI_Irecv(&A[0][0],L,MPI_DOUBLE, myrank-1, 1215,
             MPI_COMM_WORLD, &req[0]);
if(myrank!=ranksize-1)
    MPI_Isend(&A[nrow][0],L,MPI_DOUBLE, myrank+1, 1215,
             MPI_COMM_WORLD,&req[2]);
if(myrank!=ranksize-1)
    MPI_Irecv(&A[nrow+1][0],L,MPI_DOUBLE, myrank+1, 1216,
             MPI_COMM_WORLD, &req[3]);
if(myrank!=0)
    MPI_Isend(&A[1][0],L,MPI_DOUBLE, myrank-1, 1216,
             MPI_COMM_WORLD,&req[1]);
ll=4; shift=0;
if (myrank==0) {ll=2;shift=2;}
if (myrank==ranksize-1) {ll=2;}
MPI_Waitall(ll,&req[shift],&status[0]);
```



Алгоритм Якоби. MPI/OpenMP-версия

```
for(i=1; i<=nrow; i++)
{
    if (((i==1)&&(myrank==0))||((i==nrow)&&(myrank==ranksize-1))) continue;
    #pragma omp parallel for
    for(j=1; j<=L-2; j++)
        B[i-1][j] = (A[i-1][j]+A[i+1][j]+
                    A[i][j-1]+A[i][j+1])/4.;
}
}/*DO it*/
printf("%d: Time of task=%lf\n",myrank,MPI_Wtime()-t1);
MPI_Finalize ();
return 0;
}
```



Алгоритм Якоби. DVMH-версия

```
#define L 4096
#define ITMAX 100
#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
        {
            #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.;
        }
    }
    return 0;
}
```

```
}
```



DVM-система

DVM-система состоит из следующих компонент:

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

Аббревиатура DVM (**D**istributed **V**irtual **M**emory, **D**istributed **V**irtual **M**achine) отражает поддержку виртуальной общей памяти на распределенных системах

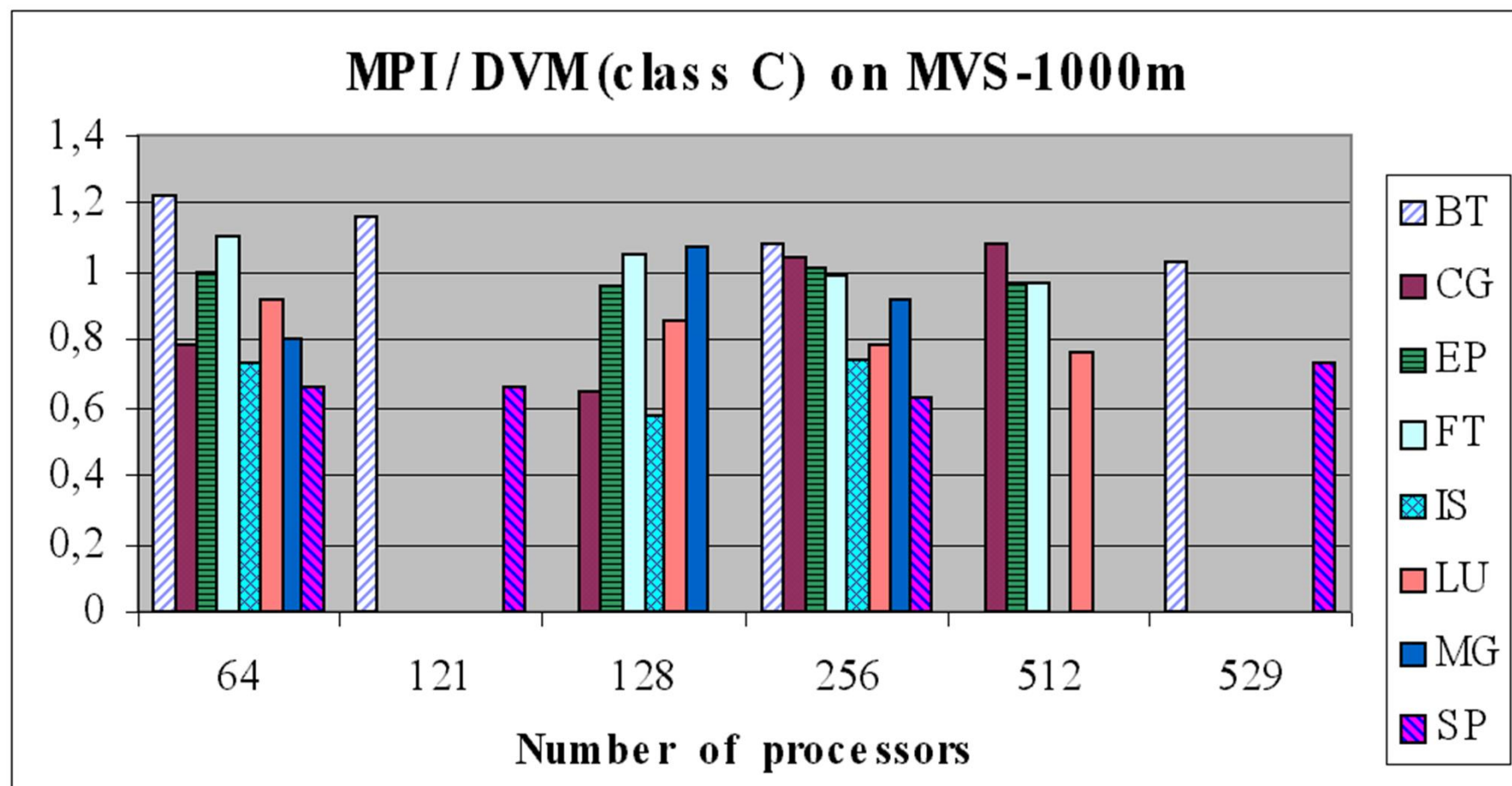


Тесты NAS Parallel Benchmarks

Тест	Характеристика теста	SEQ	MPI	DVM	MPI/SEQ	DVM/SEQ
BT	3D Навье-Стокс, метод переменных направлений	3929	5744	3991	1.46	1.02
CG	Оценка наибольшего собственного значения симметричной разреженной матрицы	1108	1793	1118	1.62	1.01
EP	Генерация пар случайных чисел Гаусса	641	670	649	1.04	1.01
FT	Быстрое преобразование Фурье, 3D спектральный метод	1500	2352	1605	1.57	1.07
IS	Параллельная сортировка	925	1218	1067	1.32	1.17
LU	3D Навье-Стокс, метод верхней релаксации	4189	5497	4269	1.31	1.02
MG	3D уравнение Пуассона, метод Multigrid	1898	2857	2131	1.50	1.12
SP	3D Навье-Стокс, Beam-Warning approximate factorization	3361	5020	3630	1.49	1.08
Σ		17551	25151	18460	1.43	1.05



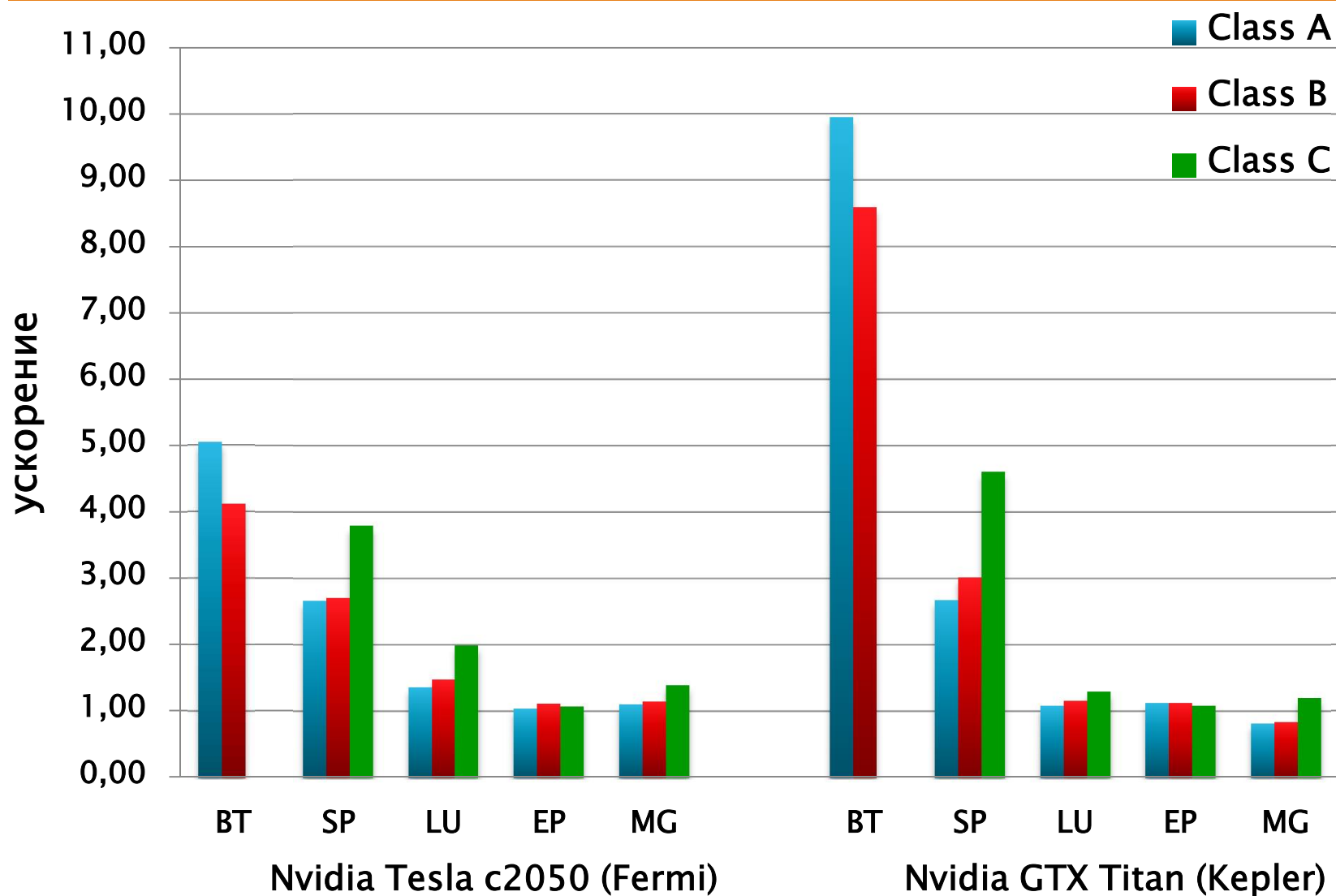
Тесты NAS Parallel Benchmarks



<http://dvm-system.org/ru/category/performance/>

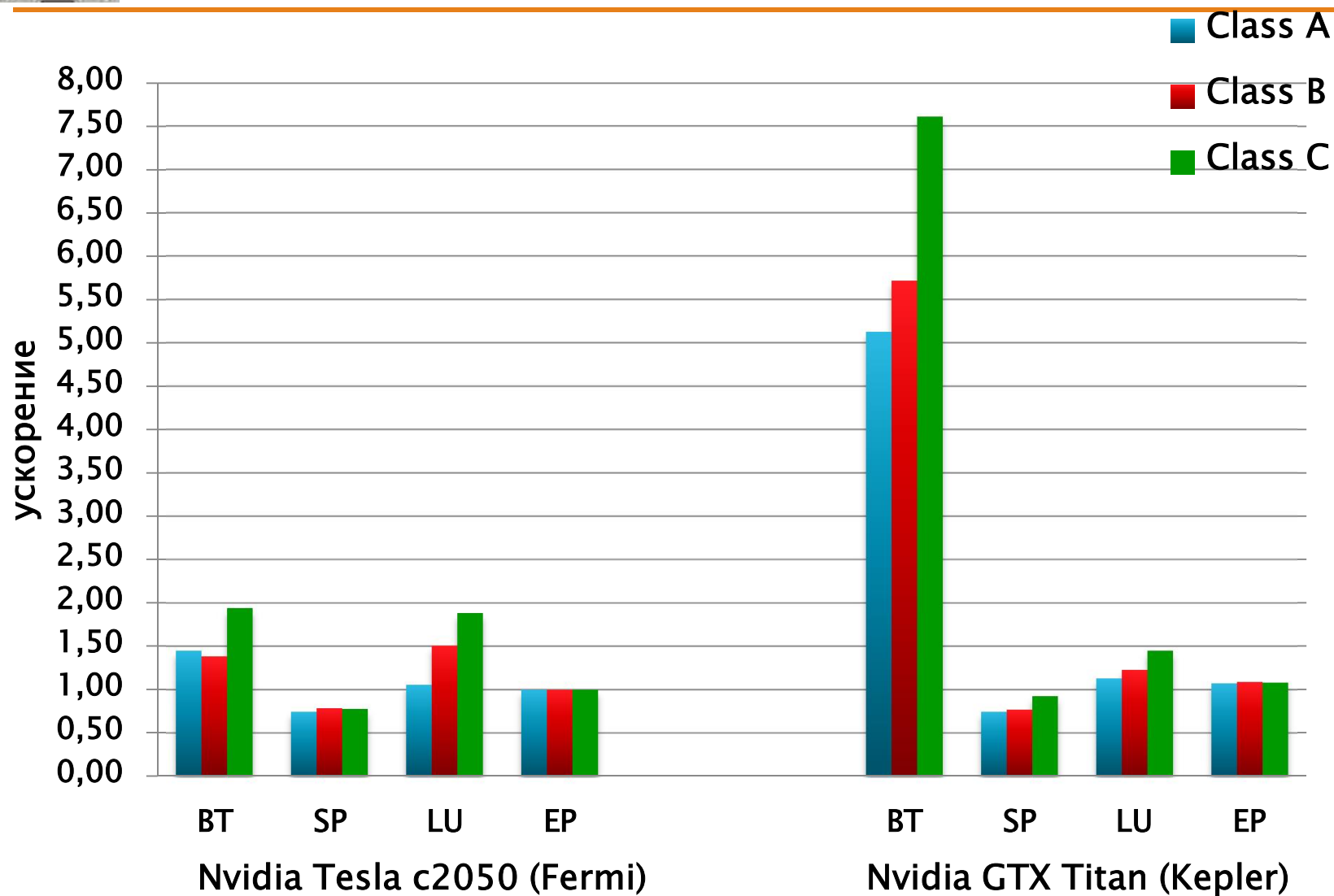


Ускорение выполнения DVMH-версий тестов NAS NPB по сравнению с OpenCL-версиями





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





Содержание

- Современные направления развития параллельных вычислительных систем
- Технологии параллельного программирования
- Система SAPFOR (System FOR Automated Parallelization)



Система SAPFOR

Средства профилирования и выделения областей распараллеливания.

Средства поиска последовательностей преобразований для автоматического устранения проблем распараллеливания.

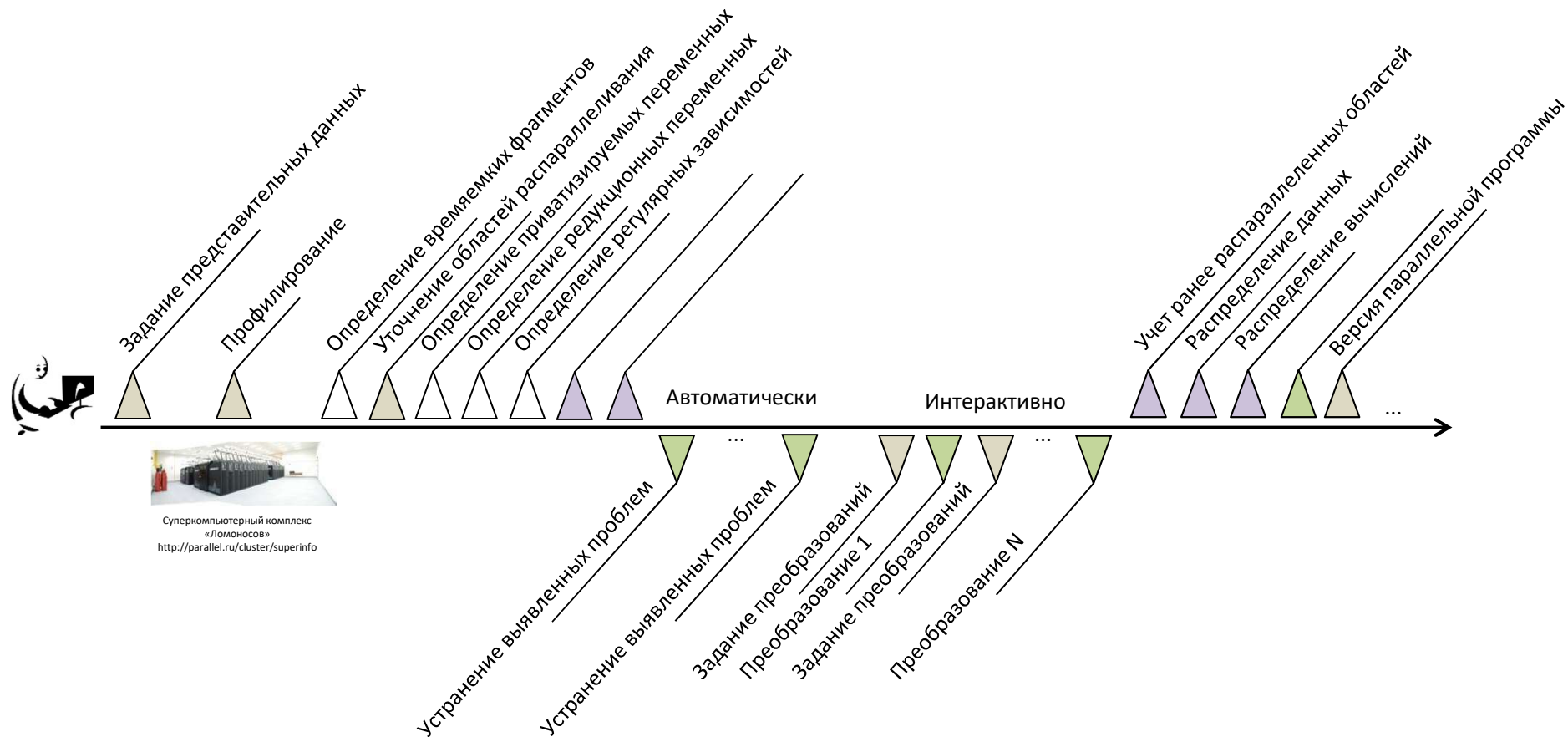


Средства автоматического выполнения наиболее часто встречающихся преобразований.

Средства интерактивного взаимодействия с пользователем в процессе распараллеливания.

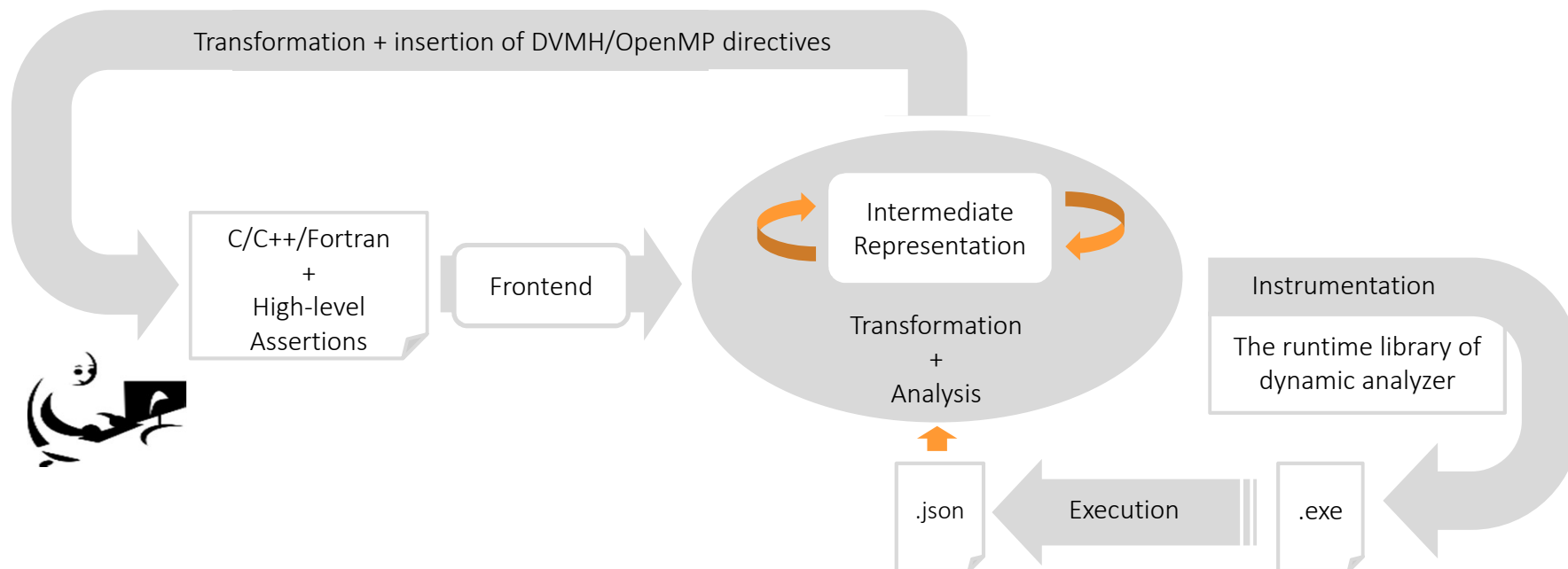


Система SAPFOR





Архитектура системы SAPFOR



- Для управления распараллеливанием используется графический пользовательский интерфейс.
- Инструменты автоматизации сборки, такие как Make, также можно использовать для выполнения анализа программ.



Система SAPFOR

Визуализатор

Проект: v2 :Очистка DVM директив Справка Компиляция и запуск Обратная связь

Анализы Преобразования Опции SAPFOR Очистка проекта

Синтаксический анализ Объявления массивов Граф циклов Граф вызовов Анализ покрытия кода Анализ кода Распределение данных

Общая Версии Анализ кода Массивы **Граф вызовов** Журнал

Циклы **Функции**

вызов binvcrhs в строке 980
цикл в строке 1016 тесная вложенность 3
цикл в строке 1017 тесная вложенность 3
цикл в строке 1018
цикл в строке 1026 тесная вложенность 4
цикл в строке 1027 тесная вложенность 3
цикл в строке 1028 тесная вложенность 2
цикл в строке 1029
цикл в строке 1037 тесная вложенность 4
цикл в строке 1038 тесная вложенность 3
цикл в строке 1039 тесная вложенность 2
цикл в строке 1040
цикл в строке 1574 тесная вложенность 2

циклов: 39

all_solve.for язык Fortran стиль: Фиксированный тип: программа

```
1015 !-----
1016 DO K = GRID_POINTS(3) - 1 - 1, 0, (-1))
1017     DO M = 1, BLOCK_SIZE
1018         DO N = 1, BLOCK_SIZE
1019             BHS(M, I, J, K) = BHS(M, I, J, K) - LHS(M, N, CC,
1020                 &, I, J, K + 1)
1021         ENDDO
1022     ENDDO
1023 ENDDO
1024 ENDDO
1025 ENDDO
1026 DO K = 1, GRID_POINTS(3) - 2
1027     DO J = 1, GRID_POINTS(2) - 2
1028         DO I = 1, GRID_POINTS(1) - 2
1029             DO M = 1, 5
```

строка 1016 из 1595 символ: 1

Предупреждения: 1 Ошибки: 0 Примечания: 4 Вывод Ошибки Распределение данных

#3001: Примечание в строке 1016: Добавлена across-зависимость к массиву 'bhs' в цикле.
#3006: Примечание в строке 1017: Неизвестная зависимость по массиву препятствует распараллеливанию данного цикла.
#3006: Примечание в строке 1018: Невозможность сопоставить обращение к массиву на запись препятствует распараллеливанию данного цикла.
#3006: Примечание в строке 1019: ANTI dependence between bhs(n,i,j,k + 1) (в строке 1019) and bhs(m,i,j,k) (в строке 1017) with unknown distance in loop в строке 1017 prevents parallelization.

C:\SAPFOR\Tests\BTv2

язык Fortran

- add.for
- adi.for
- all_solve.for
- bt.for
- error.for
- exact_rhs.for
- exact_solution.for
- header.h
- initialize.for
- npbparams.h
- print_result.for
- rhs.for
- set_constants.for
- timers.for
- verify.for
- work_lhs.h



Система SAPFOR

Визуализатор

Проект: v2 :Очистка DVM директив

Справка | Компиляция и запуск | Обратная связь

Анализы | Преобразования | Опции SAPFOR | Очистка проекта

Синтаксический анализ Объявления массивов Граф циклов Граф вызовов Анализ покрытия кода Анализ кода Распределение данных

Общая | Версии | Анализ кода | Массивы | **Граф вызовов** | Журнал

показать фильтры | Выделить всё | Отменить выделение

Разрешить распределять выделенные массивы Запретить распределять выделенные массивы

всего возможных параллельных вариантов: 0

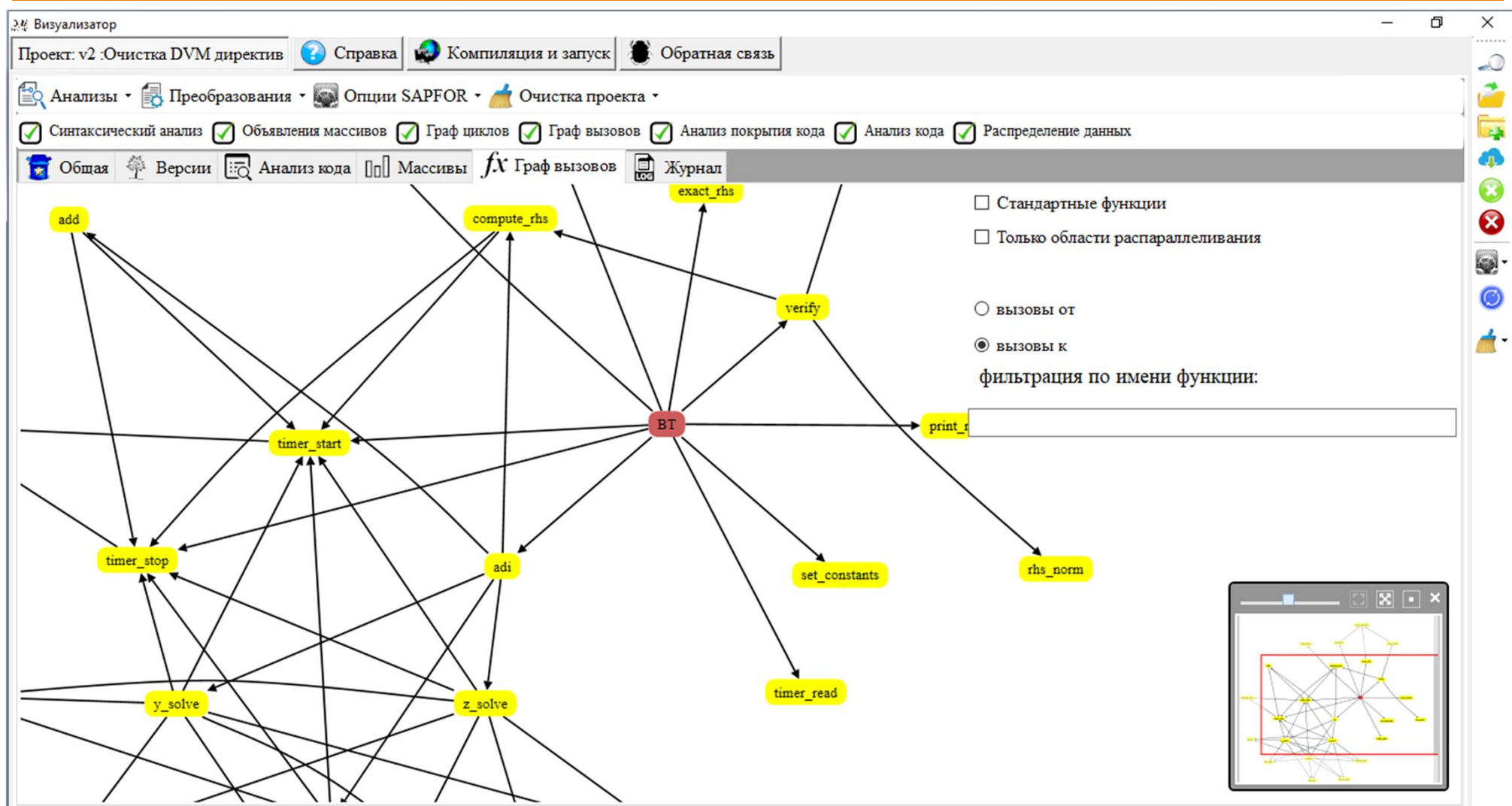
доступно: 31 разрешено распределять: 31 запрещено распределять: 0

недоступно: 18 SPF private: 9 IO private: 9

id	Имя	Область описания	Файл объявления	Размерность	Размер элемента (байт)	Область распараллеливания
5	<input checked="" type="checkbox"/> qs(0:162, 0:162, 0:161)	fields : common	показать	3	8	DEFAULT
6	<input checked="" type="checkbox"/> rho_i(0:162, 0:162, 0:161)	fields : common	показать	3	8	DEFAULT
7	<input checked="" type="checkbox"/> square(0:162, 0:162, 0:161)	fields : common	показать	3	8	DEFAULT
8	<input checked="" type="checkbox"/> forcing(1:5, 0:162, 0:162, 0:161)	fields : common	показать	4	8	DEFAULT
9	<input checked="" type="checkbox"/> u(1:5, 0:162, 0:162, 0:161)	fields : common	показать	4	8	DEFAULT
10	<input checked="" type="checkbox"/> rhs(1:5, 0:162, 0:162, 0:161)	fields : common	показать	4	8	DEFAULT
11	<input checked="" type="checkbox"/> b(1:5, 0:162, 0:162, 0:161)	fields : common	показать	4	8	DEFAULT
12	<input checked="" type="checkbox"/> bhs(1:5, 0:162, 0:162, 0:161)	fields : common	показать	4	8	DEFAULT
13	<input checked="" type="checkbox"/> cuf(0:162)	work_1d : common	показать	1	8	DEFAULT
14	<input checked="" type="checkbox"/> q(0:162)	work_1d : common	показать	1	8	DEFAULT
15	<input checked="" type="checkbox"/> ue(0:162, 1:5)	work_1d : common	показать	2	8	DEFAULT
16	<input checked="" type="checkbox"/> buf(0:162, 1:5)	work_1d : common	показать	2	8	DEFAULT
17	<input checked="" type="checkbox"/> xcref(1:5)	verify : local	показать	1	8	DEFAULT



Система SAPFOR





Система SAPFOR

Визуализатор

Проект: v2 :Очистка DVM директив

Справка | Компиляция и запуск | Обратная связь

Анализы | Преобразования | Опции SAPFOR | Очистка проекта

Синтаксический анализ | Циклы | Преобразование циклов с метками | Анализ кода | Распределение данных

Общая | SPF | Приватные переменные | Разделение циклов

Циклы | fx | Функции | Слияние циклов

главная пр | DVM | DVM директивы | фиксированный | тип: программа

вызов s | Интервалы

вызов t | Области распараллеливания

вызов i | Предобработка проекта

вызов e | Подстановка заголовочных файлов

вызов a | вызов i | вызов t | вызов t

вызов mod в строке 134

вызов adi в строке 138

вызов timer_stop в строке 140

вызов timer_read в строке 141

вызов verify в строке 142

объявлений: 1 | вызовов: 16

```
58 IF (.NOT. (TIMERON)) GOTO 999
59 DO I = 1, T_LAST
60   TRECS(I) = TIMER_READ (I)
61 ENDDO
62 IF (TMAX .EQ. 0.0) TMAX = 1.0
63 WRITE (UNIT = *, FMT = 800)
64 800 FORMAT (' SECTION Time (secs)')
65 DO I = 1, T_LAST
66   WRITE (UNIT = *, FMT = 810) T_NAMES(I), TRECS(I), TRECS(:
67   &/ TMAX
68   IF (I .EQ. T_RHS) THEN
69     T = TRECS(T_RHSX) + TRECS(T_RHSY) + TRECS(T_RHSZ)
70     WRITE (UNIT = *, FMT = 820) 'sub-rhs', T, T * 100. /
71     T = TRECS(T_RHS) - T
72     WRITE (UNIT = *, FMT = 820) 'rest-rhs', T, T * 100. /
```

строка 159 из 186 | символ: 1

Предупреждения: 10 | Ошибки: 0 | Примечания: 6 | Вывод | Ошибки | Распределение данных

#1016: Примечание в строке 133: Невозможно вычислить количество итераций данного цикла, информация о количестве итераций для всех остальных циклов в области распараллеливания 'DEFAULT' будет проигнорирована.

#3006: Примечание в строке 133: Операторы ввода/вывода препятствуют распараллеливанию данного цикла.

#3006: Примечание в строке 159: Обращение к нераспределенному массиву на запись препятствует распараллеливанию данного цикла.

#1030: Примечание в строке 165: Добавлена приватная переменная 't' для этого цикла.

#3006: Примечание в строке 165: Операторы ввода/вывода препятствуют распараллеливанию данного цикла.

C:\SAPFOR\Tests\BTv2

язык Fortran

- add.for
- adi.for
- all_solve.for
- bt.for
- error.for
- exact_rhs.for
- exact_solution.for
- header.h
- initialize.for
- npbparams.h
- print_result.for
- rhs.for
- set_constants.for
- timers.for
- verify.for
- work_lhs.h



Система SAPFOR

Визуализатор

Проект: v2 :Очистка DVM директив

Справка | Компиляция и запуск | Обратная связь

Анализы | Преобразования | Опции SAPFOR | Очистка проекта

Синтаксический анализ
 Объявления массивов
 Граф циклов
 Граф вызовов
 Анализ покрытия кода
 Анализ кода
 Распределение данных

Общая | Версии | Анализ кода | Массивы | **Граф вызовов** | Журнал

Родительская | Оригинальная

ВТ : исходная
 m1 : копия от 13:22:32 26 ноября 2019 г.
 v1 : Сохранение текущего проекта
 m2 : копия от 13:22:49 26 ноября 2019 г.
 v2 : Очистка DVM директив
 m1 : копия от 13:59:17 26 ноября 2019 г.
 v1 : Устранение конфликтов областей

Параллельные варианты | Сравнение

Добавить вариант | Добавить все возможные варианты

распределить измерения	область	i	j	k	l
dvmh_temp0 (i, j, k, l)	DEFAULT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

всего вариантов:

не выбрано: 0
 выбрано: 16

области	шаблоны	P.Rem	P.Sha	P.Red	P.Acr	Parallel	Remote	Realig
<input checked="" type="checkbox"/> DEFAULT	dvmh_temp0(*,*,*,BLOCK)	0	17	2	0	57	0	62
<input checked="" type="checkbox"/> DEFAULT	dvmh_temp0(*,BLOCK,*,*)	0	17	2	0	57	0	68



Система SAPFOR

Визуализатор

Проект: v2 : Очистка DVM директив

Справка | Компиляция и запуск | Обратная связь

Анализы | Преобразования | Опции SAPFOR | Очистка проекта

Синтаксический анализ Объявления массивов Граф циклов Граф вызовов Анализ покрытия кода Анализ кода Распределение данных

Общая | Версии | Анализ кода | Массивы | Граф вызовов | Журнал

Родительская | Оригинальная | Параллельные варианты | Сравнение

- BT : исходная
 - m1 : копия от 13:22:32 26 ноября 2019 г.
 - v1 : Сохранение текущего проекта
 - m2 : копия от 13:22:49 26 ноября 2019 г.
 - v2 : Очистка DVM директив
 - m1 : копия от 13:59:17 26 ноября 2019 г.
 - v1 : Устранение конфликтов областей
 - m2 : копия от 14:24:41 26 ноября 2019 г.
 - p1 : dvmh_temp0(*,*,BLOCK,*)

```
p1 (dvmh_temp0(*,*,BLOCK,*)) header.h
0 10 20 30 40
57 c
58 c to improve cache performance, grid c
59 c for even number sizes only.
60 !DVM$ ALIGN b(iEX1,iEX2,iEX3,iEX4) WITH
61 !DVM$ ALIGN bhs(iEX1,iEX2,iEX3,iEX4) WIT
62 !DVM$&4)
63 !DVM$ ALIGN forcing(iEX1,iEX2,iEX3,iEX4)
64 !DVM$&,iEX4)
65 !DVM$ ALIGN qs(iEX1,iEX2,iEX3) WITH dvmh
66 !DVM$ ALIGN rho_i(iEX1,iEX2,iEX3) WITH c
67 !DVM$ ALIGN rhs(iEX1,iEX2,iEX3,iEX4) WIT
68 !DVM$&4)
69 !DVM$ ALIGN square(iEX1,iEX2,iEX3) WITH
70 !DVM$ ALIGN u(iEX1,iEX2,iEX3,iEX4) WITH
71 !DVM$ ALIGN us(iEX1,iEX2,iEX3) WITH dvmh
72 !DVM$ ALIGN vs(iEX1,iEX2,iEX3) WITH dvmh
73 !DVM$ ALIGN ws(iEX1,iEX2,iEX3) WITH dvmh
74 !DVM$ DYNAMIC b,us,vs,bhs,square,qs,u,ws
75 !DVM$ SHADOW qs(1:1,1:1,1:1)
76 !DVM$ SHADOW rho_i(1:1,1:1,1:1)
77 !DVM$ SHADOW square(1:1,1:1,1:1)
78 !DVM$ SHADOW u(0:0,1:1,2:2,1:1)
79 !DVM$ SHADOW us(1:1,1:1,1:1)
80 !DVM$ SHADOW vs(1:1,1:1,1:1)

v1 (Устранение конфликтов областей) header.h
0 10 20 30 40
57 c
58 c to improve cache performance, grid di
59 c for even number sizes only.
60 - !DVM$ ALIGN b(iEX1,iEX2,iEX3,iEX4) WITH
61 - !DVM$ ALIGN bhs(iEX1,iEX2,iEX3,iEX4) WI
62 - !DVM$&4)
63 - !DVM$ ALIGN forcing(iEX1,iEX2,iEX3,iEX4)
64 - !DVM$&,iEX4)
65 - !DVM$ ALIGN qs(iEX1,iEX2,iEX3) WITH dvm
66 - !DVM$ ALIGN rho_i(iEX1,iEX2,iEX3) WITH
67 - !DVM$ ALIGN rhs(iEX1,iEX2,iEX3,iEX4) WI
68 - !DVM$&4)
69 - !DVM$ ALIGN square(iEX1,iEX2,iEX3) WITH
70 - !DVM$ ALIGN u(iEX1,iEX2,iEX3,iEX4) WITH
71 - !DVM$ ALIGN us(iEX1,iEX2,iEX3) WITH dvm
72 - !DVM$ ALIGN vs(iEX1,iEX2,iEX3) WITH dvm
73 - !DVM$ ALIGN ws(iEX1,iEX2,iEX3) WITH dvm
74 - !DVM$ DYNAMIC b,us,vs,bhs,square,qs,u,w
75 - !DVM$ SHADOW qs(1:1,1:1,1:1)
76 - !DVM$ SHADOW rho_i(1:1,1:1,1:1)
77 - !DVM$ SHADOW square(1:1,1:1,1:1)
78 - !DVM$ SHADOW u(0:0,1:1,2:2,1:1)
79 - !DVM$ SHADOW us(1:1,1:1,1:1)
80 - !DVM$ SHADOW vs(1:1,1:1,1:1)
```

Опции



Вопросы, замечания?

Спасибо!



<http://dvm-system.org>





Литература

- ❑ DVM-система. <http://www.keldysh.ru/dvm>
<http://www.dvm-system.org>
- ❑ OpenMP Application Program Interface Version 3.0, May 2008.
<http://www.openmp.org/mp-documents/spec30.pdf>
- ❑ MPI: A Message-Passing Interface Standard Version 2.2, September 2009. <http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf>

- ❑ Антонов А.С. Параллельное программирование с использованием технологии OpenMP: Учебное пособие.-М.: Изд-во МГУ, 2009.
<http://parallel.ru/info/parallel/openmp/OpenMP.pdf>
- ❑ Антонов А.С. Параллельное программирование с использованием технологии MPI: Учебное пособие.-М.: Изд-во МГУ, 2004.
http://parallel.ru/tech/tech_dev/MPI/mpibook.pdf
- ❑ Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВ-Петербург, 2002.
- ❑ Э. Таненбаум, М. ван Стеен. Распределенные системы. Принципы и парадигмы. – СПб. Питер, 2003



Автор

Бахтин Владимир Александрович, кандидат физико-математических наук,
заведующий сектором Института прикладной математики им. М.В. Келдыша РАН,
доцент кафедры системного программирования факультета ВМК, МГУ им. М. В.
Ломоносова

bakhtin@keldysh.ru



Инициализация и завершение MPI программ

Первой вызываемой функцией MPI должна быть функция:

```
int MPI_Init ( int *argc, char ***argv )
```

Для инициализации среды выполнения MPI-программы. Параметрами функции являются количество аргументов в командной строке и текст самой командной строки.

Последней вызываемой функцией MPI обязательно должна являться функция:

```
int MPI_Finalize (void)
```

[Обратно](#)



Определение количества и ранга процессов

Определение количества процессов в выполняемой параллельной программе осуществляется при помощи функции:

```
int MPI_Comm_size ( MPI_Comm comm, int *size ).
```

Для определения ранга процесса используется функция:

```
int MPI_Comm_rank ( MPI_Comm comm, int *rank ).
```

[Обратно](#)



Неблокирующие обмены данными между процессорами

Для передачи сообщения процесс-отправитель должен выполнить функцию:

```
int MPI_Isend(void *buf, int count, MPI_Datatype type, int dest,  
             int tag, MPI_Comm comm, MPI_Request *request),
```

где

- `buf` - адрес буфера памяти, в котором располагаются данные отправляемого сообщения,
- `count` - количество элементов данных в сообщении,
- `type` - тип элементов данных пересылаемого сообщения,
- `dest` - ранг процесса, которому отправляется сообщение,
- `tag` - значение-тег, используемое для идентификации сообщений,
- `comm` - коммуникатор, в рамках которого выполняется передача данных.

[Обратно](#)

Для приема сообщения процесс-получатель должен выполнить функцию:

```
int MPI_Irecv(void *buf, int count, MPI_Datatype type, int source,  
             int tag, MPI_Comm comm, MPI_Status *status, MPI_Request *request),
```

где

- `buf`, `count`, `type` - буфер памяти для приема сообщения, назначение каждого отдельного параметра соответствует описанию в `MPI_Send`,
- `source` - ранг процесса, от которого должен быть выполнен прием сообщения,
- `tag` - тег сообщения, которое должно быть принято для процесса,
- `comm` - коммуникатор, в рамках которого выполняется передача данных,
- `status` - указатель на структуру данных с информацией о результате выполнения операции приема данных.



MPI_Waitall

Ожидание завершения всех операций обмена осуществляется при помощи функции:

```
int MPI_Waitall(  
int count,  
    MPI_Request array_of_requests[],  
    MPI_Status array_of_statuses[])
```

[Обратно](#)



MPI_Cart_create

Создание декартовой топологии (решетки) в MPI:

```
int MPI_Cart_create(MPI_Comm oldcomm, int ndims, int *dims, int *periods,  
int reorder, MPI_Comm *cartcomm),
```

где:

- `oldcomm` - исходный коммуникатор,
- `ndims` - размерность декартовой решетки,
- `dims` - массив длины `ndims`, задает количество процессов в каждом измерении решетки,
- `periods` - массив длины `ndims`, определяет, является ли решетка периодической вдоль каждого измерения,
- `reorder` - параметр допустимости изменения нумерации процессов,
- `cartcomm` - создаваемый коммуникатор с декартовой топологией процессов.

[Обратно](#)



MPI_Cart_shift

Функция:

int MPI_Cart_shift(MPI_Comm comm, int dir, int disp, int *source, int *dst)

для получения номеров посылающего(source) и принимающего (dst) процессов в декартовой топологии коммутатора (comm) для осуществления сдвига вдоль измерения dir на величину disp.

[Обратно](#)



MPI_Card_coords

Определение декартовых координат процесса по его рангу:

```
int MPI_Card_coords(MPI_Comm comm,int rank,int ndims,int *coords),
```

где:

- `comm` - коммуникатор с топологией решетки,
- `rank` - ранг процесса, для которого определяются декартовы координаты,
- `ndims` - размерность решетки,
- `coords` - возвращаемые функцией декартовы координаты процесса.

[Обратно](#)



MPI_Type_vector

Для снижения сложности в MPI предусмотрено несколько различных способов конструирования производных типов:

- *Непрерывный* способ позволяет определить непрерывный набор элементов существующего типа как новый производный тип,
- *Векторный* способ обеспечивает создание нового производного типа как набора элементов существующего типа, между элементами которого существуют регулярные промежутки по памяти. При этом, размер промежутков задается в числе элементов исходного типа,
- *Индексный* способ отличается от векторного метода тем, что промежутки между элементами исходного типа могут иметь нерегулярный характер,
- *Структурный* способ обеспечивает самое общее описание производного типа через явное указание карты создаваемого типа данных.

```
int MPI_Type_vector(int count, int blocklen, int stride, MPI_Data_type oldtype,  
MPI_Datatype *newtype),
```

где

- `count` - количество блоков,
- `blocklen` - размер каждого блока,
- `stride` - количество элементов, расположенных между двумя соседними блоками
- `oldtype` - исходный тип данных,
- `newtype` - новый определяемый тип данных.

[Обратно](#)



MPI_Type_commit

Перед использованием производный тип должен быть объявлен при помощи функции:

```
int MPI_Type_commit (MPI_Datatype *type )
```

При завершении использования производный тип должен быть аннулирован при помощи функции:

```
int MPI_Type_free (MPI_Datatype *type ).
```

[Обратно](#)