Лекция 6. Библиотеки быстрых вычислений

Инструменты разработки быстрых программ

18 декабря 2017 г.

Введение

Рассматриваемые библиотеки

- FFTW 3.3.7 (The Fast Fourier Transform in the West);
- ScaLAPACK (Scalable Linear Algebra Package);
- Intel MKL (Math Kernel Library).

Вычисление одномерного комплексного преобразования

Одномерное комплексное преобразование Фурье

$$y_k = \sum_{j=0}^{n-1} x_j \, e^{-k\,j\,\frac{2\pi i}{n}}$$

$$z_k = \sum_{j=0}^{n-1} y_j \, e^{k\,j\,\frac{2\pi i}{n}}$$

Одномерное комплексное преобразование

Пример (C)

```
#include <complex.h>    // C99

#include <fftw3.h>

// ...

int main()
{
  fftw_complex *pInp = fftw_alloc_complex(N);
  fftw_complex *pOut = fftw_alloc_complex(N);
  fftw_plan plan = fftw_plan_dft_1d(
    N, pInp, pOut, FFTW_FORWARD, FFTW_ESTIMATE);
```

Одномерное комплексное преобразование

Пример (C)

```
  fill_data(N, pInp);
  fftw_execute(plan);
  process_data(N, pOut);
  fftw_destroy_plan(plan);
  fftw_free(pInp);
  fftw_free(pOut);
  fftw_cleanup();
}    // main()
```

Одномерное комплексное преобразование (окончание)

Пример (Сборка, C)

```
$ gcc fftw_one_complex.c -lfftw3 -lm
$                        -lfftw3f
$                        -lfftw3l
```

Определение (C)

```
void *fftw_malloc(size_t n);
double *fftw_alloc_real(size_t n);
fftw_complex *fftw_alloc_complex(size_t n);
void fftw_free(void *pv);
//  fftwf_*, fftwl_*
```

Функции одномерного комплексного преобразования

Определение (C)

```
fftw_plan fftw_plan_dft_1d(
  int n0,
  fftw_complex *pInp, fftw_complex *pOut,
  int nSign, unsigned uFlags);

fftw_plan fftw_plan_dft_2d(    // 3d
  int n0, int n1, /* ... */);

fftw_plan fftw_plan_dft(
  int nRank, const int *pnDims, /* ... */);
```

nSign	FFTW_FORWARD, FFTW_BACKWARD
uFlags	FFTW_ESTIMATE, FFTW_MEASURE, FFTW_PATIENT, FFTW_EXHAUSTIVE

Таблица 1: возможные значения параметров

Функции использования планов

Определение (C)

```
void fftw_execute(const fftw_plan plan);
void fftw_destroy_plan(fftw_plan plan);
```

Функции нескольких комплексных преобразований

Определение (C)

```
fftw_plan fftw_plan_many_dft(
  int nRank, const int *pnDims, int nHowMany,
  fftw_complex *pInp, const int *pINEmbed,
  int nIStride, int nIDist,
  fftw_complex *pOut, const int *pnONEmbed,
  int nOStride, int nODist,
  int nSign, unsigned uFlags);
```

Расположение элементов в памяти

$x_j^{(l)} \sim$ `pInp[j * nIStride + l * nIDist]`

Вычисление многомерного комплексного преобразования

Многомерное комплексное преобразование Фурье

$$y_{k_1,k_2,\ldots,k_d} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} x_{j_1,j_2,\ldots,j_d}\, \omega_1^{-k_1 j_1}\, \omega_2^{-k_2 j_2} \ldots \omega_d^{-k_d j_d}$$

$$z_{k_1,k_2,\ldots,k_d} = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \cdots \sum_{j_d=0}^{n_d-1} y_{j_1,j_2,\ldots,j_d}\, \omega_1^{k_1 j_1}\, \omega_2^{k_2 j_2} \ldots \omega_d^{k_d j_d}$$

где $\omega_s \stackrel{\text{def}}{=} e^{\frac{2\pi i}{n_s}}$ — s-й корень уравнения $z^{n_s} = -1, \quad \forall s \in [1, d]_{\mathbb{N}}$

Трёхмерное комплексное преобразование

Пример (C)

```
// ...

int main()
{
  fftw_complex *pData = (fftw_complex *) fftw_malloc(
    sizeof (fftw_complex) * M * N * P);
  fftw_plan plan = fftw_plan_dft_3d(
    M, N, P, pData, pData,
    FFTW_FORWARD, FFTW_ESTIMATE);
```

Трёхмерное комплексное преобразование

Пример (C, окончание)

```
  fill_data(M, N, P, pData);
  process_data(M, N, P, pData);
  fftw_execute(plan);
  process_data(M, N, P, pData);
  fftw_destroy_plan(plan);
  fftw_free(pData);
  fftw_cleanup();
}    // main()
```

Одномерное вещественное преобразование

Пример (C)

```
// ...

int main()
{
  double *pInp = fftw_alloc_real(N);
  double *pOut = fftw_alloc_real(N);
  fftw_plan plan = fftw_plan_r2r_1d(
    N, pInp, pOut, FFTW_R2HC, FFTW_ESTIMATE);
  fill_data(N, pInp);
  fftw_execute(plan);
  // ...
}
```

Функции вещественного в вещественное преобразования

Определение (C)

```
fftw_plan fftw_plan_r2r_1d(
  int n, double *pdInp, double *pdOut,
  fftw_r2r_kind nKind, unsigned uFlags);

// fftw_plan_r2r_2d(), fftw_plan_r2r_3d(),
// fftw_plan_r2r()
```

Расположение элементов вещественного преобразования

Расположение элементов вещественного преобразования

$$r_0, r_1, r_2, \ldots, r_{\lfloor \frac{n}{2} \rfloor}, i_{\lfloor \frac{n+1}{2} \rfloor - 1}, \ldots, i_2, i_1$$

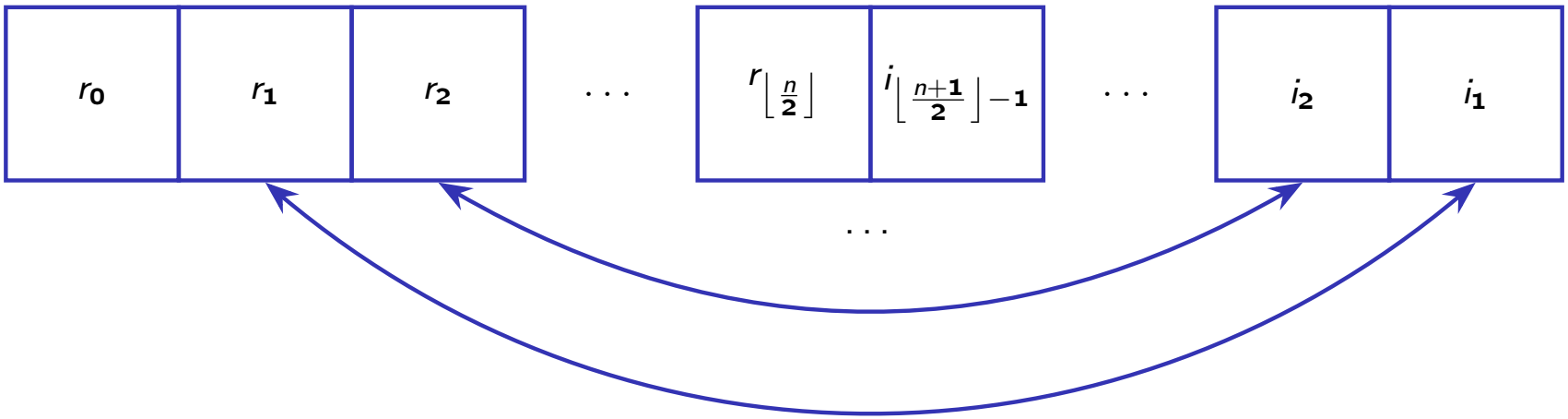

Рис. 1: Схема расположения вещественных и мнимых частей преобразования

Одномерное вещественное преобразование (окончание)

Пример (C)

```
void process_data(size_t uSize, double *pdData)
{
  size_t i, uSizeU = (uSize + 1) / 2, uSizeH = uSize / 2;
  char szFormat[] = "%4lu (% 8.3lf, % 8.3lf)\n";
  printf(szFormat, 0, pdData[0], 0);
  //
  for (i = 1; i < uSizeU; ++ i)
    printf(szFormat, i, pdData[i], pdData[uSize - i]);
  //
  if (0 == uSize % 2)
    printf(szFormat, uSizeH, pdData[uSizeH], 0);
}
```

Функции многомерного вещественного преобразования

Определение (C)

```
fftw_plan fftw_plan_dft_r2c_1d(
//                          c2r
//                             _2d(
//                             _3d(
//                            (
  int n0,
  double *pdInp, fftw_complex *pOut,
  unsigned uFlags);
```

Массивы для вещественного преобразования

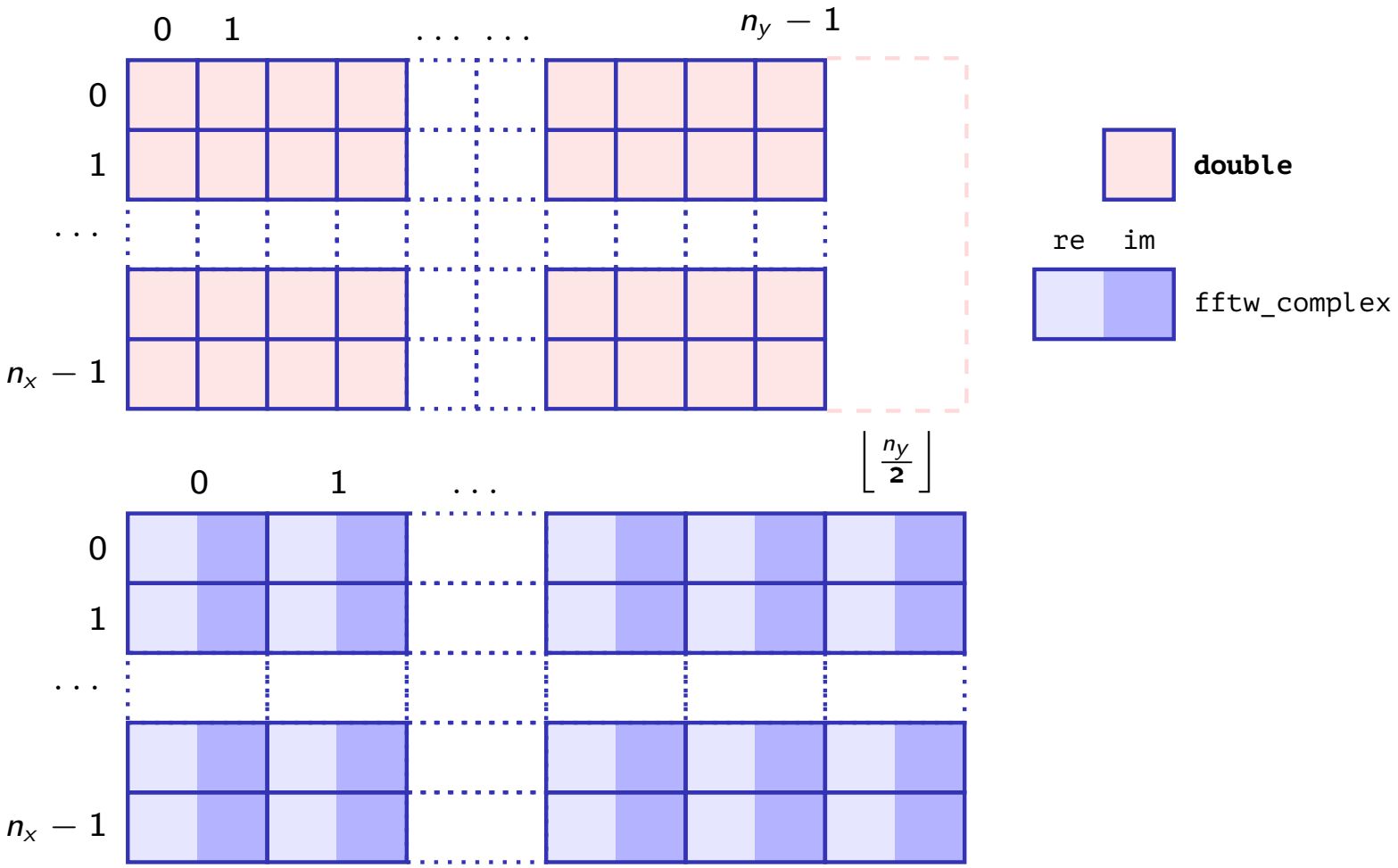

Рис. 2: Формат данных для многомерного вещественного преобразования

Массивы для вещественного преобразования

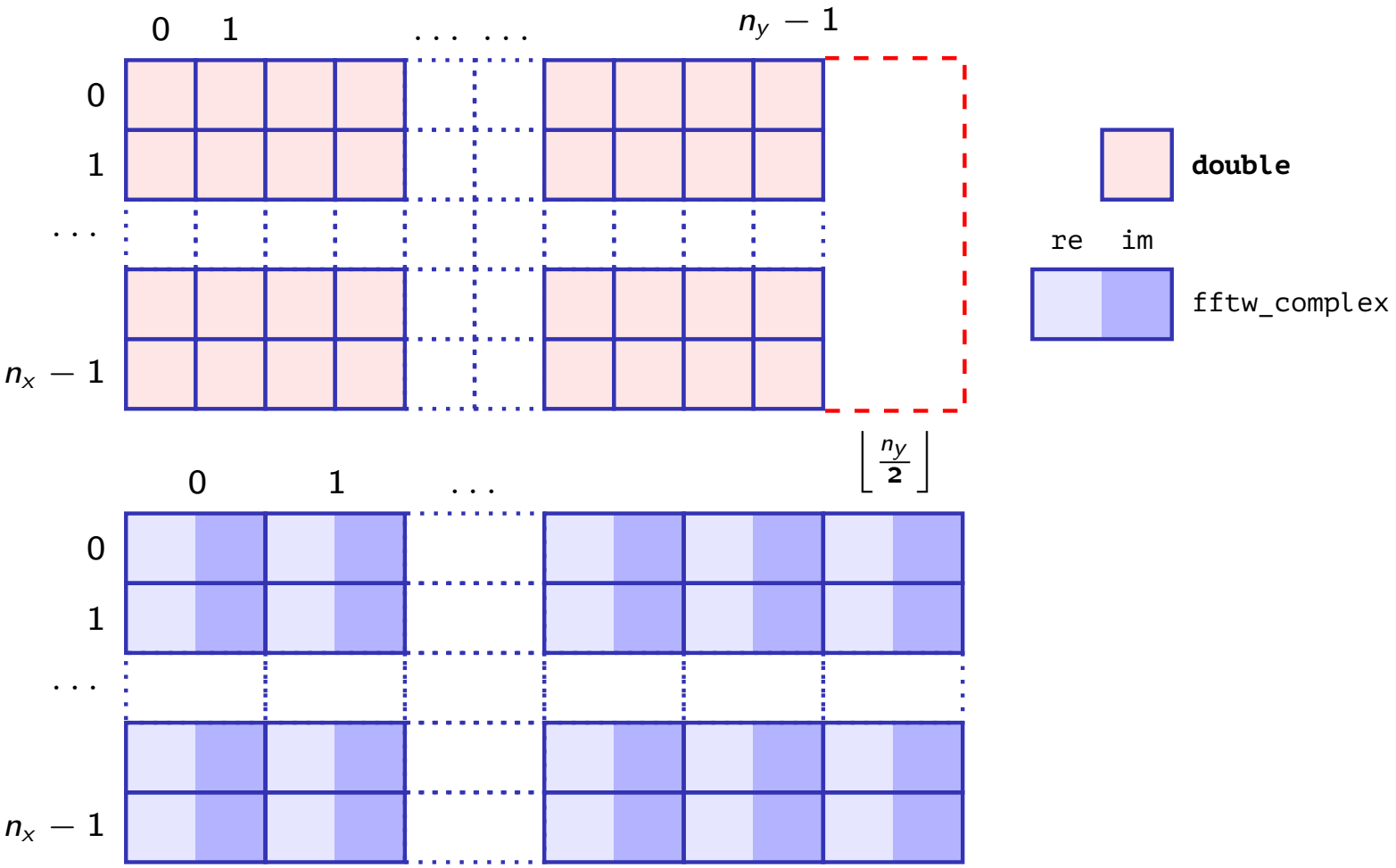

Рис. 2: Формат данных для многомерного вещественного преобразования

Данные многомерного вещественного преобразования

Пример (C)

```c
#include <fftw3.h>

// ...

int main()
{
  double aaInp[M][N];
  fftw_complex aaOut[M][N / 2];
  double aaInpOut[M][2 * (N / 2 + 1)];
  // ...
}
```

Параллельные реализации

Варианты параллельной реализации библиотеки

- SSE, SSE2, AVX, Altivec, MIPS PS
- POSIX Threads (или OpenMP).
- MPI (многомерные комплексные/вещественные, одномерные комплексные преобразования).

Сборка многопоточных версий программы

Пример (Сборка, cc, POSIX Threads)

```
gcc fftw3_threads_one_complex.c \
  -lfftw3_threads -lfftw3 -lm -lgomp -pthread
```

Пример (Сборка, gcc, OpenMP)

```
gcc fftw3_threads_one_complex.c \
  -lfftw3_threads -lfftw3 -lm -fopenmp -lgomp
```

Многопоточное одномерное преобразование

Пример (C)

```
#include <fftw3.h>
#include <omp.h>

// ...

int main()
{
  fftw_complex *pInp = fftw_alloc_complex(N);
  fftw_complex *pOut = fftw_alloc_complex(N);
  fftw_plan plan;
```

Многопоточное одномерное преобразование (окончание)

Пример (C, окончание)

```
  int nThreads = omp_get_num_procs();
  if (fftw_init_threads() == 0)
    return -1;
  fftw_plan_with_nthreads(nThreads);
  // ...
  fftw_cleanup_threads();
}    // main()
```

Функции комплексного преобразования MPI

Определение (C)

```
fftwnd_mpi_plan fftw2d_mpi_create_plan(
  MPI_Comm nComm, int nX, int nY,
  fftw_direction nDirection, int nFlags);

void fftwnd_mpi(
  fftwnd_mpi_plan plan, int nFields,
  fftw_complex *pLocal, fftw_complex *pWork,
  fftwnd_mpi_output_order nOutputOrder);
```

Возможные значения параметров

nOutputOrder — FFTW_NORMAL_ORDER, FFTW_TRANSPOSED_ORDER

Функции комплексного преобразования MPI (окончание)

Определение (C)

```
void fftwnd_mpi_local_sizes(
  fftwnd_mpi_plan plan,
  int *pnLocalNX,
  int *pnLocalXStart,
  int *pnLocalNYAfterTranspose,
  int *pnLocalYStartAfterTranspose,
  int *pnTotalLocalSize);
```

Преобразование с использованием MPI

Пример (C)

```
#include <stdlib.h>
#include <fftw_mpi.h>
/* ... */

int main(int nArgC, char *apszArgV[])
{
  int nSize, nRank;
  int nLocalNX, nLocalXStart, nLocalNYAfterTrans,
    nLocalYStartAfterTrans, nTotalLocalSize;
  fftw_complex *pIn = NULL, *pOut = NULL, *pLocal = NULL, *pWork = NULL;
  int *pnCounts = NULL, *pnDisplacements = NULL;
  fftwnd_mpi_plan plan;
```

Преобразование с использованием MPI (продолжение)

Пример (C, продолжение)

```
//
MPI_Init(&nArgC, &apszArgV);
//
MPI_Comm_size(MPI_COMM_WORLD, &nSize);
MPI_Comm_rank(MPI_COMM_WORLD, &nRank);
//
if (0 == nRank)
{
  pIn = (fftw_complex *) calloc(M * N, sizeof (fftw_complex));
  fill_data(M, N, pIn);
  pnCounts = (int *) calloc(nSize, sizeof (int));
  pnDisplacements = (int *) calloc(nSize, sizeof (int));
}    // if (0 == nRank)
```

Преобразование с использованием MPI (продолжение)

Пример (C, продолжение)

```
//
plan = fftw2d_mpi_create_plan(
  MPI_COMM_WORLD, M, N, FFTW_FORWARD,
  FFTW_ESTIMATE /* | FFTW_IN_PLACE */);
//
fftwnd_mpi_local_sizes(
  plan, &nLocalNX, &nLocalXStart, &nLocalNYAfterTrans,
  &nLocalYStartAfterTrans, &nTotalLocalSize);
//
pLocal =
  (fftw_complex *) calloc(nTotalLocalSize, sizeof (fftw_complex));
pWork =
  (fftw_complex *) calloc(nTotalLocalSize, sizeof (fftw_complex));
```

Преобразование с использованием MPI (продолжение)

Пример (C, продолжение)

```
//
nLocalNX *= 2 * N;
nLocalXStart *= 2 * N;
//
MPI_Gather(
  &nLocalNX, 1, MPI_INT,
  pnCounts, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Gather(
  &nLocalXStart, 1, MPI_INT,
  pnDisplacements, 1, MPI_INT, 0, MPI_COMM_WORLD);
//
```

Преобразование с использованием MPI (продолжение)

Пример (C, продолжение)

```
MPI_Scatterv(
  pIn, pnCounts, pnDisplacements, MPI_DOUBLE,
  pLocal, nLocalNX, MPI_DOUBLE, 0, MPI_COMM_WORLD);
//
fftwnd_mpi(plan, 1, pLocal, pWork, FFTW_NORMAL_ORDER);
fftwnd_mpi_destroy_plan(plan);
//
if (0 == nRank)
  pOut = (fftw_complex *) calloc(M * N, sizeof (fftw_complex));
//
MPI_Gatherv(
  pLocal, nLocalNX, MPI_DOUBLE,
  pOut, pnCounts, pnDisplacements, MPI_DOUBLE, 0, MPI_COMM_WORLD);
```

Преобразование с использованием MPI (окончание)

Пример (C, окончание)

```
  //
  if (0 == nRank)
    process_data(M, N, pOut);
  //
  MPI_Finalize();
  free(pIn);
  free(pOut);
  free(pLocal);
  free(pWork);
  free(pnCounts);
  free(pnDisplacements);
}    // main()
```

Сборка и запуск версий MPI программы

Пример (Сборка)

```
#!/bin/sh

mpicc ./fftw_mpi_two_complex.c -lfftw_mpi -lfftw -lm
```

Пример (Запуск)

```
#!/bin/sh

mpirun -nolocal -np 64 -machinefile ./mpd.hosts ./a.out
```

ScaLAPACK

Состав

- BLAS (ATLAS);
- LAPACK;
- BLACS;
- MPI.

Возможности

Состав

- Решение систем линейных уравнений;
- Нахождение собственных значений;
- Метод наименьших квадратов для линейных регрессионных моделей;
- Сингулярное разложение.

Intel MKL

Состав

- BLAS, LAPACK;
- FFT;
- VML (Vector Mathematical Functions);
- VSL (Vector Statistical Functions).

Сборка программы MKL ScaLAPACK

Пример (Сборка)

```
#!/bin/sh

ifort ./scalapack_example1.f \
  -lmkl_scalapack_lp64 -lmpi -lmkl_blacs_openmpi_lp64 \
  -lmkl_intel_lp64 -lmkl_lapack -lmkl_core -lmkl_intel_thread -liomp5 \
  -L/share/mpi/openmpi-1.4.2/lib \
  -L/opt/intel/Compiler/11.0/074/mkl/lib/em64t \
  -L/opt/intel/Compiler/11.0/074/lib/intel64
```