# ОБЗОР ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ

Любая достаточно ушедшая вперед технология неотличима от чуда.

*А. Кларк*

ФПК "Параллельные вычислительные технологии"

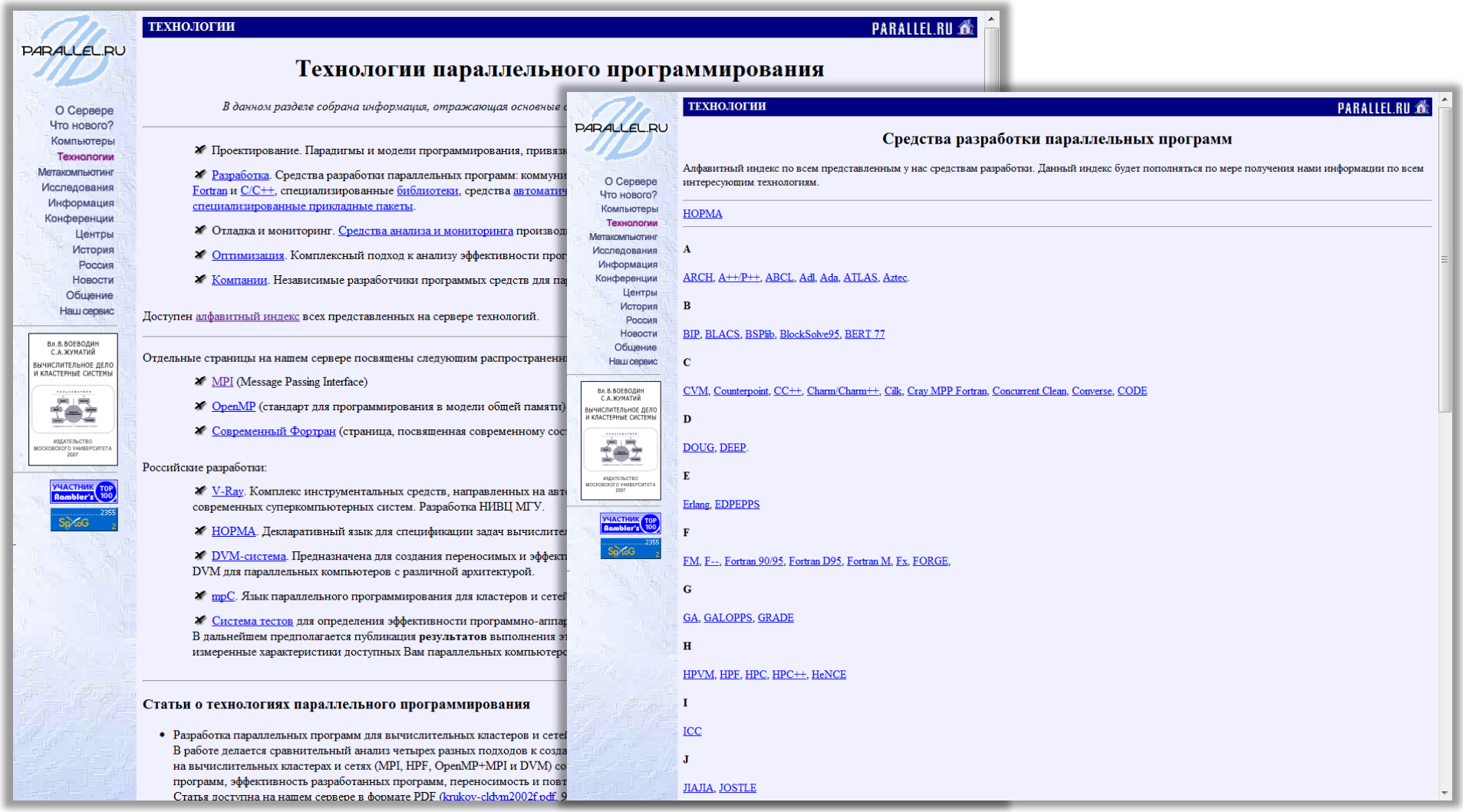
# Содержание



- Критерии выбора технологии параллельного программирования
- Классификация и обзор технологий параллельного программирования

# Технологии параллельного программирования





ТЕХНОЛОГИИ PARALLEL.RU

## Технологии параллельного программирования

*В данном разделе собрана информация, отражающая основные с*

- Проектирование. Парадигмы и модели программирования, привязк
- Разработка. Средства разработки параллельных программ: коммуни Fortran и C/C++, специализированные библиотеки, средства автоматич специализированные прикладные пакеты.
- Отладка и мониторинг. Средства анализа и мониторинга производ
- Оптимизация. Комплексный подход к анализу эффективности прог
- Компании. Независимые разработчики программных средств для па

Доступен алфавитный индекс всех представленных на сервере технологий.

Отдельные страницы на нашем сервере посвящены следующим распространенн

- MPI (Message Passing Interface)
- OpenMP (стандарт для программирования в модели общей памяти)
- Современный Фортран (страница, посвященная современному сос

Российские разработки:

- V-Ray. Комплекс инструментальных средств, направленных на авт современных суперкомпьютерных систем. Разработка НИВЦ МГУ.
- HOPMA. Декларативный язык для спецификации задач вычислител
- DVM-система. Предназначена для создания переносимых и эффекти DVM для параллельных компьютеров с различной архитектурой.
- mpC. Язык параллельного программирования для кластеров и сетей
- Система тестов для определения эффективности программно-аппар В дальнейшем предполагается публикация **результатов** выполнения эт измеренные характеристики доступных Вам параллельных компьютерс

### Статьи о технологиях параллельного программирования

- Разработка параллельных программ для вычислительных кластеров и сете В работе делается сравнительный анализ четырех разных подходов к созда на вычислительных кластерах и сетях (MPI, HPF, OpenMP+MPI и DVM) со программ, эффективность разработанных программ, переносимость и повт Статья доступна на нашем сервере в формате PDF (krukov-cldvm2002f.pdf, 9

ТЕХНОЛОГИИ PARALLEL.RU

## Средства разработки параллельных программ

Алфавитный индекс по всем представленным у нас средствам разработки. Данный индекс будет пополняться по мере получения нами информации по всем интересующим технологиям.

HOPMA

**A**

ARCH, A++/P++, ABCL, Adl, Ada, ATLAS, Aztec.

**B**

BIP, BLACS, BSPlib, BlockSolve95, BERT 77

**C**

CVM, Counterpoint, CC++, Charm/Charm++, Cilk, Cray MPP Fortran, Concurrent Clean, Converse, CODE

**D**

DOUG, DEEP.

**E**

Erlang, EDPEPPS

**F**

FM, F--, Fortran 90/95, Fortran D95, Fortran M, Fx, FORGE,

**G**

GA, GALOPPS, GRADE

**H**

HPVM, HPF, HPC, HPC++, HeNCE

**I**

ICC

**J**

JIAJIA, JOSTLE

# Критерии выбора технологии



1. Эффективность создаваемых программ
   - Технологии, дающие неэффективные программы, не нужны!
2. Быстрота создания программ
   - Простота освоения технологии специалистами других предметных областей
3. Сохранение эффективности программ
   - Гарантия сохранения эффективности программ при их переносе на другую аппаратно-программную платформу

# Классификация технологий

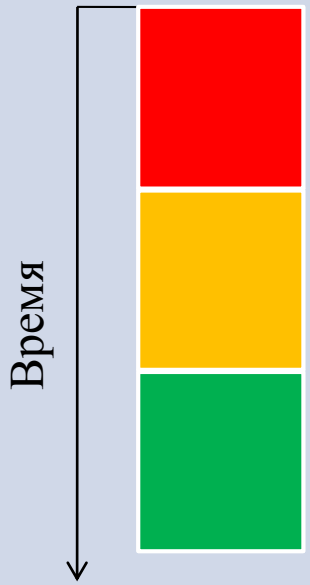


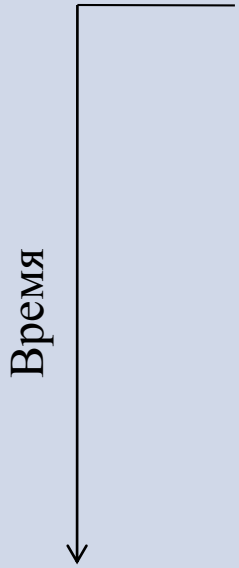
- Распараллеливающие компиляторы
- Параллельные языки программирования
- Параллельные расширения последовательных языков программирования
- Высокоуровневые коммуникационные библиотеки
- Параллельные библиотеки
- Инструментальные системы разработки параллельных программ
- Специализированные прикладные пакеты

# Распараллеливающие компиляторы



| Текст программы на последовательном языке | Исполняемый код последовательного компилятора | Исполняемый код распараллеливающего компилятора |
|---|---|---|

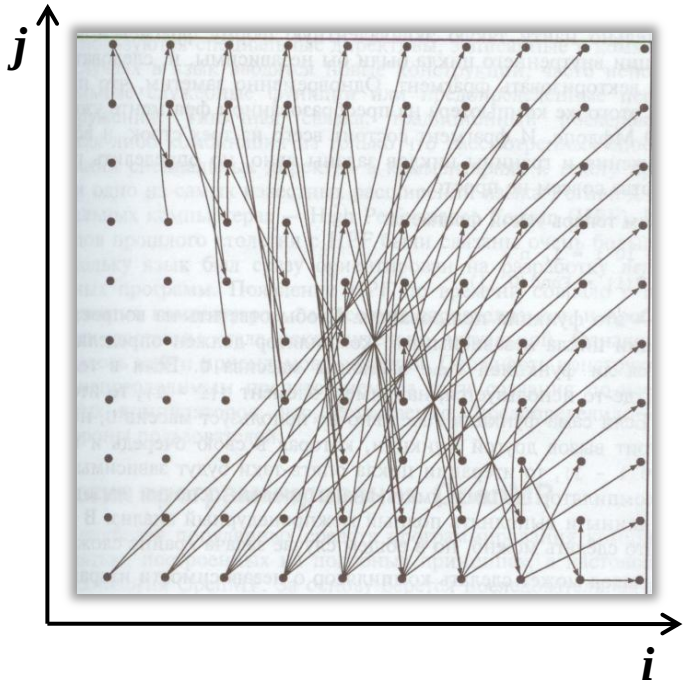



# Распараллеливающие компиляторы



```
for (i=1; i<=n; i++)
 for (j=1; j<=n; j++)
  A[i+j]=A[2*n-i-j+1]*q+p
```

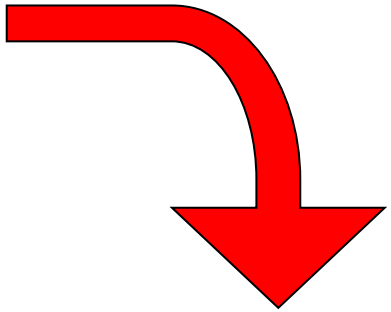


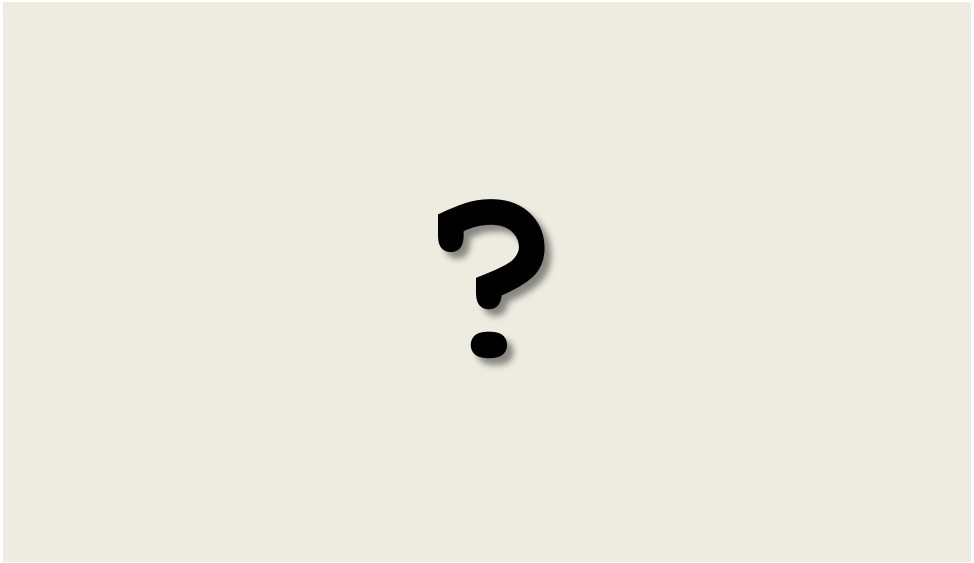

```
for (i=1; i<=n; i++) {
 for (j=1; j<=n-i; j++)
  A[i+j]=A(2*n-i-j+1)*q+p;
 for (j=n-i+1; j<=n; j++)
  A[i+j]=A(2*n-i-j+1)*q+p;
}
```

# Распараллеливающие компиляторы



```
for (i=1; i<=n; i++)
 A[i]=UserFunc(A, B[i]);
```





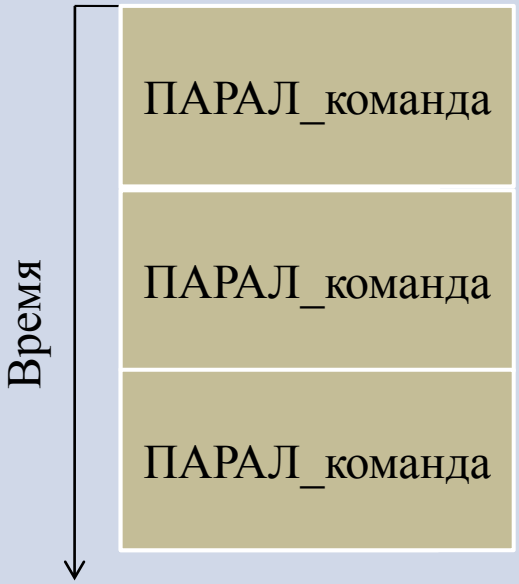
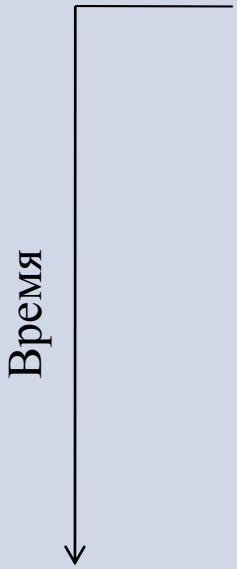
# Классификация технологий



- Распараллеливающие компиляторы
- **Параллельные языки программирования**
- Параллельные расширения последовательных языков программирования
- Высокоуровневые коммуникационные библиотеки
- Параллельные библиотеки
- Инструментальные системы разработки параллельных программ
- Специализированные прикладные пакеты

# Параллельные языки



| Текст программы на параллельном языке | Исполняемый код |
| --- | --- |
|  |  |

# Параллельные языки

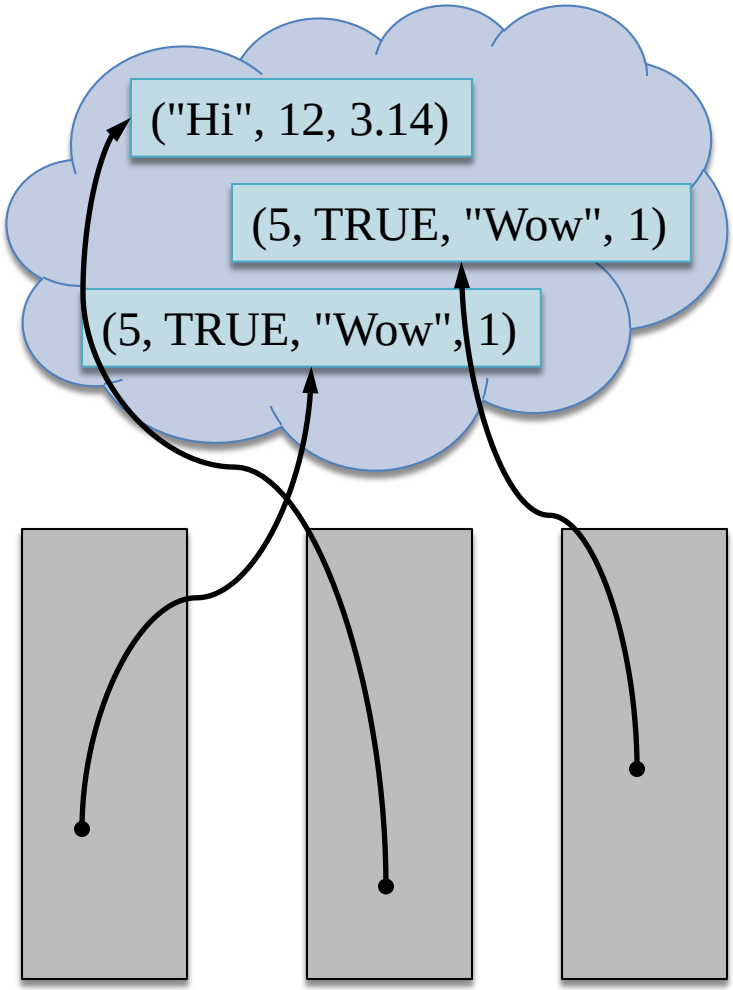


- Параллельные языки
  - HOPMA, ABCL, Adl, Ada, Concurrent Clean, MC#, Erlang, Linda, Modula-3, NESL, Occam, Orca, Parallaxis, Phantom, Sisal, SR, ZPL и др.

# Язык и модель Linda



- Разработаны в 1980 гг. в Йельском университете (США).
- Параллельная программа – множество параллельных процессов, каждый из которых работает как последовательная программа.
- Процессы имеют доступ к общей памяти – *пространству кортежей*.
- *Кортеж* – упорядоченная последовательность значений.
- Процессы взаимодействуют друг с другом неявно, через пространство кортежей с помощью операций "поместить", "забрать", "скопировать" кортеж.
- Программа считается завершенной, если все процессы завершились или заблокированы.



# Функции Linda



- `OUT(<кортеж>)`
  - Поместить кортеж в пространство кортежей.
  - Если такой кортеж уже имеется, то создается дубликат.
  - Вызывающий процесс не блокируется.
  - Примеры
    - `out("myTuple", 1);`
    - `out(FALSE, 3.14, );`

# Функции Linda



- `IN(<кортеж>)`
  - Получить указанный кортеж и удалить его из пространства кортежей.
  - Если параметру соответствует несколько кортежей, то случайным образом выбирается один из них.
  - Вызывающий процесс блокируется, пока соответствующий кортеж не появится в пространстве кортежей.
  - Примеры
    - `in("myTuple", 1);`
    - ```
      int i,j;
      in("myTuple",1,formal i,?j);
      ```
      ```

# Функции Linda



- `READ(<кортеж>)`
  - Получить указанный кортеж из пространства кортежей (не удаляя его).
  - Если параметру соответствует несколько кортежей, то случайным образом выбирается один из них.
  - Вызывающий процесс блокируется, пока соответствующий кортеж не появится в пространстве кортежей.
  - Примеры
    - ```
      int i;
      float j;
      char * s;
      read(?s, formal i, ?j);
      ```
      ```

# Функции Linda



- `EVAL(<кортеж>)`
  - Поместить кортеж в пространство кортежей.
  - Если такой кортеж уже имеется, то создается дубликат.
  - Вызывающий процесс не блокируется.
  - Для вычисления поля кортежа, которое содержит обращение к какой-либо функции, порождается параллельный процесс.
  - Функция не ожидает завершения порожденного процесса.
  - Поля кортежа вычисляются в произвольном порядке.
  - Примеры
    - `eval("myTuple",myFunc1(a,b,c),TRUE,myFunc1(i,j,k));`

# Классификация технологий



- Распараллеливающие компиляторы
- Параллельные языки программирования
- **Параллельные расширения последовательных языков программирования**
- Высокоуровневые коммуникационные библиотеки
- Параллельные библиотеки
- Инструментальные системы разработки параллельных программ
- Специализированные прикладные пакеты

# Параллельные расширения



- Дополнение последовательного языка программирования параллельными операторами или директивами компилятора
  - Параллельные расширения и диалекты языка Fortran
    - Fortran-DVM, Cray MPP Fortran, F--, Fortran 90/95, Fortran D95, Fortran M, Fx, **HPF**, Opus, Vienna Fortran и др.
  - Параллельные расширения и диалекты языков C/C++
    - C-DVM, A++/P++, CC++, Charm/Charm++, Cilk, HPC, HPC++, **C/OpenMP** Maisie, Mentat, mpC, MPC++, Parsec, pC++, sC++, uC++ и др.

# Язык High Performance FORTRAN



- HPF – расширение языка FORTRAN, которые дают компилятору информацию для оптимизации выполнения программы на многопроцессорном (многоядерном) компьютере.
- Примеры директив
  - `!HPF$ PROCESSORS (n)`
    - Определение количества процессоров, которые могут использоваться программой.
  - `!HPF$ DISTRIBUTE (BLOCK) ONTO procs :: переменные`
    - Блочное распределение данных массива список переменных по процессорам (распределение равными блоками).
  - `ALIGN переменные1(i) WITH переменные2(i+1)`
    - Установка связи между распределением двух массивов. Для всех значений переменной i элемент массива список переменных1(i) должен быть размещен в памяти того же процессора, что и элемент массива список переменных2(i).
  - `FORALL (i=1:1000) переменные1(i)=переменныхе(i)`
    - Определение набора операторов, которые могут выполняться параллельно.

# Параллельное расширение C/OpenMP



| Последовательный язык C | Параллельное расширение C/OpenMP |
|---|---|

```
int Fact(int N)
{
  int  i, F;
  F = 1;
  for (i = 1; i<=N; i++)
    F *=  i;
  return F;
}
int N;
void main (void) {
  printf("Введите число: ');
  scanf(&N);
  printf("%d! = %d", N, Fact(N));
}
```

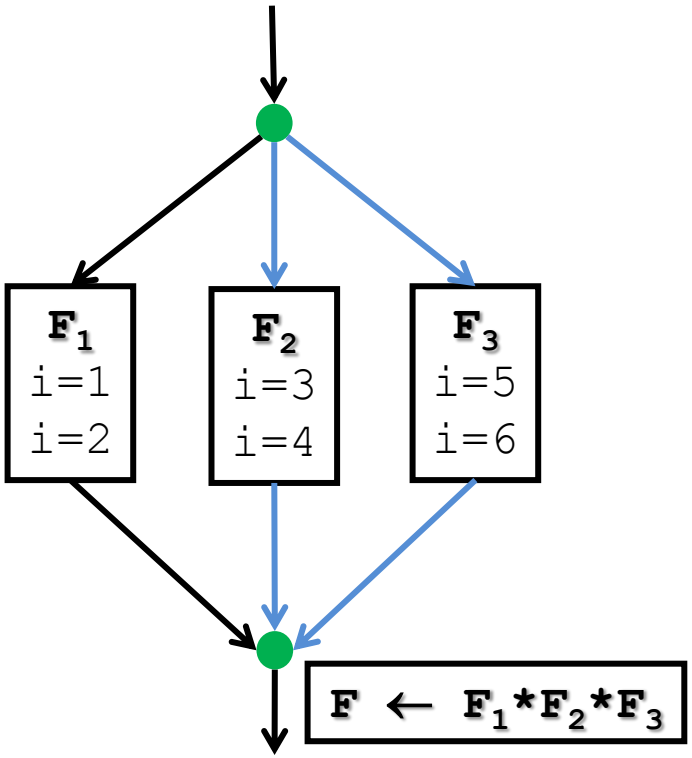
```
int Fact(int N)
{
  int  i, F;
  F = 1;
#pragma omp parallel for
reduction(*:F)
  for (i = 1; i<=N; i++)
    F *=  i;
  return F;
}
int N;
void main (void) {
  printf("Введите число: ');
  scanf(&N);
  printf("%d! = %d", N, Fact(N));
}
```

# Параллельное расширение C/OpenMP



## Параллельное расширение C/OpenMP

```
int Fact(int N)
{
  int  i, F;
  F = 1;
#pragma omp parallel for reduction(*:F)
  for (i = 1; i<=N; i++)
    F *=  i;
  return F;
}
int N;
void main (void) {
  printf("Введите число: ');
  scanf(&N);
  printf("%d! = %d", N, Fact(N));
}
```



# Классификация технологий



- Распараллеливающие компиляторы
- Параллельные языки программирования
- Параллельные расширения последовательных языков программирования
- **Высокоуровневые коммуникационные библиотеки**
- Параллельные библиотеки
- Инструментальные системы разработки параллельных программ
- Специализированные прикладные пакеты

# Интерфейсы параллельного программирования



- Программирование на стандартных и широко распространенных языках программирования с использованием высокоуровневых коммуникационных библиотек и интерфейсов (API) для организации межпроцессного взаимодействия.
- Коммуникационные библиотеки и интерфейсы
  - ACE, ARCH, BIP, BLACS, BSPlib, CVM, Counterpoint, FM, Gala, GA, HPVM, ICC, JIAJIA, KELP, LPARX, **MPI**, MPL, OOMPI, OpenMP, **POSIX Threads**, P4, Para++, Phosphorus, PVM, Quarks, ROMIO, ShMem, SVMlib, TOOPS, Windows Threads

# MPI-программа

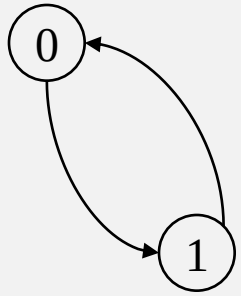


```
#include "mpi.h" // Подключение библиотеки

int main (int argc, char *argv[])
{
// Здесь код без использования MPI функций

  MPI_Init(&argc, &argv); // Инициализация выполнения

// Здесь код, реализующий обмены

  MPI_Finalize(); // Завершение

// Здесь код без использования MPI функций

 return 0;
}
```

# Пример MPI-программы



```
#include "mpi.h"
#include <stdio.h>
int main(int argc, char *argv[])
{
  int numtasks, rank, dest, src, rc, tag=777;
  char inmsg, outmsg='x';
  MPI_Status Stat;

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD, &numtasks);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  if (rank == 0) {
    dest = 1; src = 1;
    rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    rc = MPI_Recv(&inmsg, 1, MPI_CHAR, src, tag, MPI_COMM_WORLD, &Stat);
} else
  if (rank == 1) {
    dest = 0; src = 0;
    rc = MPI_Recv(&inmsg, 1, MPI_CHAR, src, tag, MPI_COMM_WORLD, &Stat);
    rc = MPI_Send(&outmsg, 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  }
  MPI_Finalize();
}
```



# Пример программы с POSIX Threads



```
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 4

void *hello (void *arg) {
      printf("Hello Thread\n");
}

main() {
  pthread_t tid[NUM_THREADS];
  for (int i=0; i<NUM_THREADS; i++)
    pthread_create(&tid[i], NULL, hello, NULL);

  for (int i=0; i<NUM_THREADS; i++)
    pthread_join(tid[i], NULL);
}
```

# Классификация технологий



- Распараллеливающие компиляторы
- Параллельные языки программирования
- Параллельные расширения последовательных языков программирования
- Высокоуровневые коммуникационные библиотеки
- **Параллельные библиотеки**
- Инструментальные системы разработки параллельных программ
- Специализированные прикладные пакеты

# Специализированные параллельные библиотеки

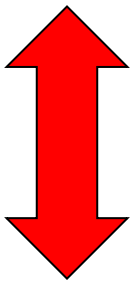


- Библиотеки заранее распараллеленных процедур для решения различных научно-технических задач (линейная алгебра, сеточные методы, методы Монте-Карло, генетические алгоритмы, рендеринг изображений, …)
  - ATLAS, Aztec, BlockSolve95, Distributed Parallelization at CWP, DOUG, GALOPPS, JOSTLE, NAMD, P-Sparslib, PIM, ParMETIS, PARPACK, PBLAS, PETSc, PGAPack, PLAPACK, ScaLAPACK, SPRNG и др.

# Параллельная библиотека Intel MKL



```
for (i=0; i<N; i++) {
 for (j=0; j<N; j++) {
  for (k=0; k<N; k++)
   C[i][j]+=A[i][k]*B[k][j];
 }
}
```



```
for (i=0; i<N; i++ ) {
 for (j=0; j<N; j++ )
  C[i][j]=cblas_ddot(N, &A[i], incx, &B[0][j], incy);
}
```

# Классификация технологий



- Распараллеливающие компиляторы
- Параллельные языки программирования
- Параллельные расширения последовательных языков программирования
- Высокоуровневые коммуникационные библиотеки
- Параллельные библиотеки
- **Инструментальные системы разработки параллельных программ**
- Специализированные прикладные пакеты

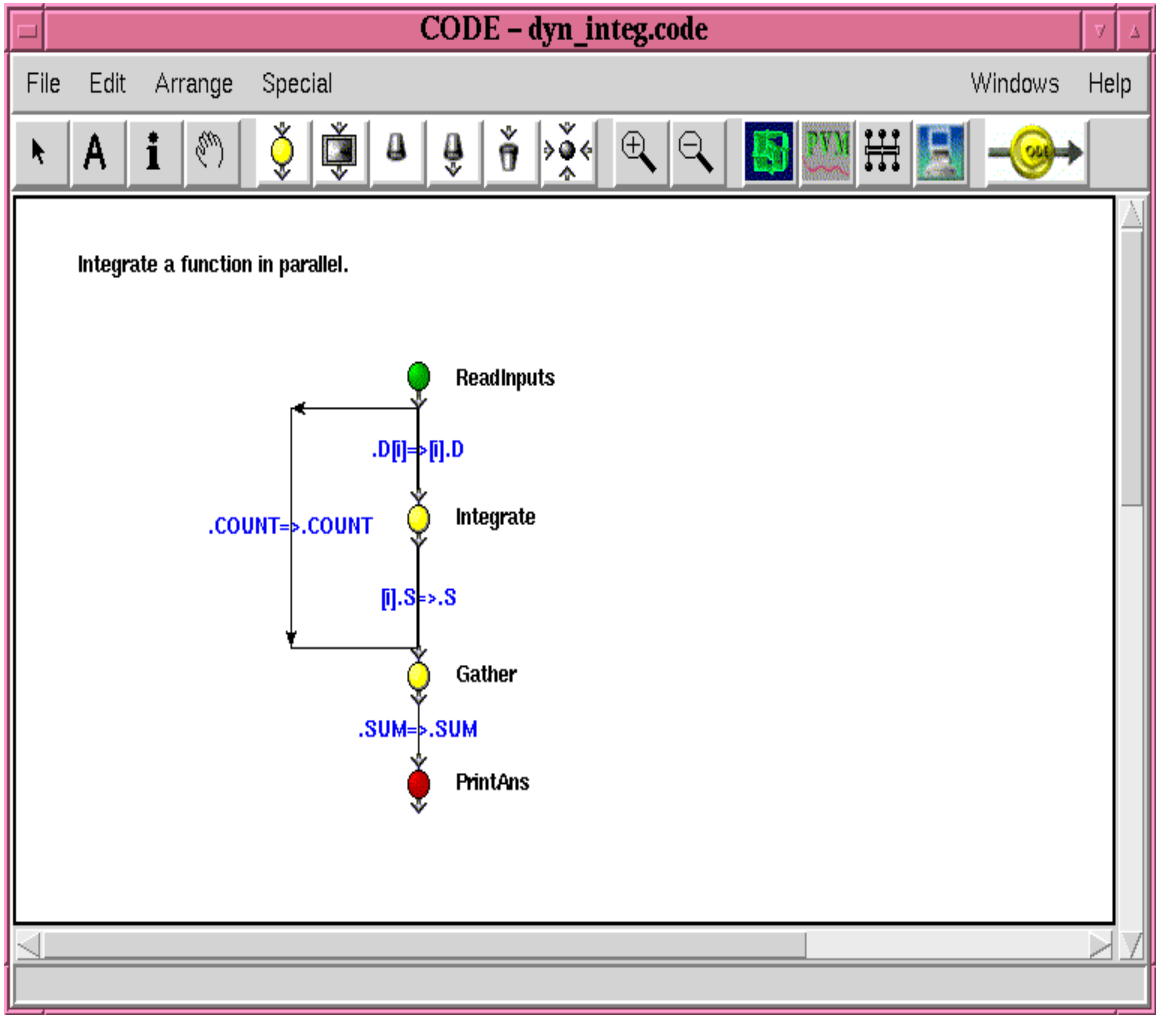
# Инструментальные системы разработки



- Интегрированные среды прототипирования, разработки и отладки параллельных программ
  - **CODE**, Converse, DEEP, EDPEPPS, GRADE, HeNCE, Reactor, TRAPPER и др.
- Средства распознавания параллелизма в алгоритмах, средства автоматического и полуавтоматического распараллеливания последовательных программ
  - BERT 77, FORGE, KAP, PIPS, VAST, **V-Ray** и др.

# Инструментальная система CODE






- Параллельная программа = граф
  - вершины – последовательные участки
  - дуги – пересылки данных.
- Последовательные участки могут быть написаны на любом языке, для пересылок используется MPI.

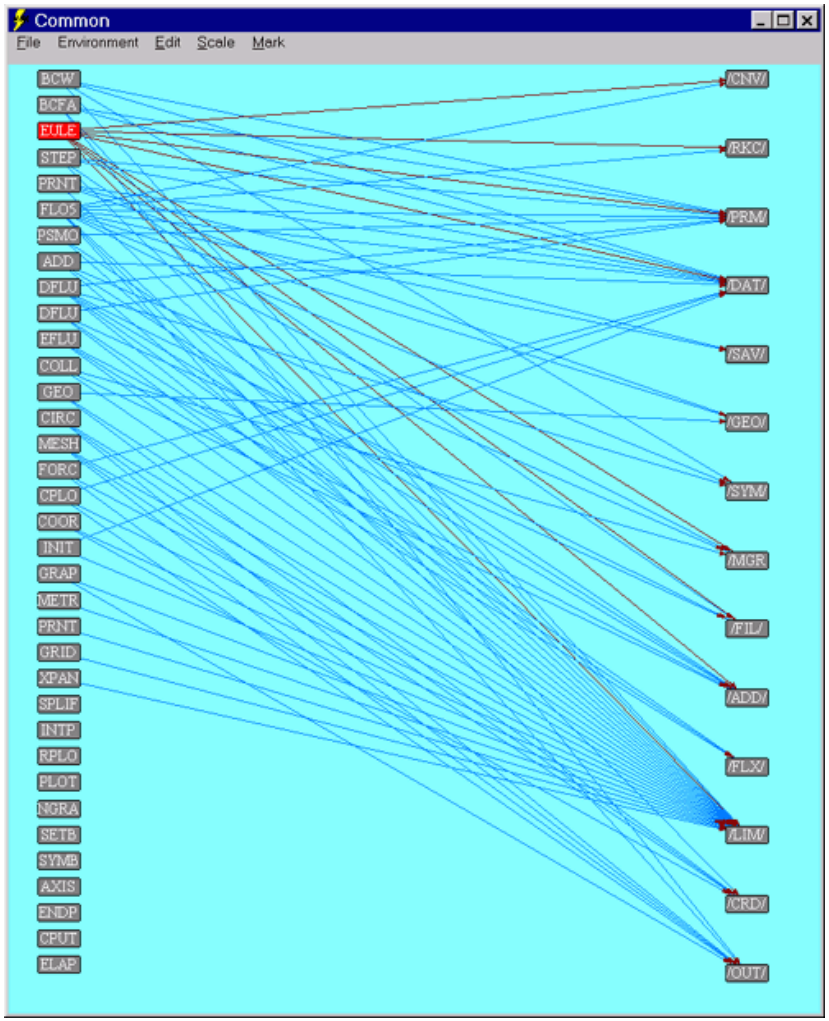
# Пакет V-Ray



- V-Ray – комплекс инструментальных средств, направленных на автоматизацию создания и оптимизацию параллельных Fortran-программ для современных суперкомпьютерных систем. Разработка НИВЦ МГУ.
- Основные возможности:
  - Макроанализ – граф вызовов (порядок вызова процедур проекта), граф вложенности циклов в процедурах, граф использования общей памяти и др.
  - Микроанализ – иерархический граф управления, определение параллельных циклов, и др.

# V-Ray: граф использования общей памяти





# Классификация технологий



- Распараллеливающие компиляторы
- Параллельные языки программирования
- Параллельные расширения последовательных языков программирования
- Высокоуровневые коммуникационные библиотеки
- Параллельные библиотеки
- Инструментальные системы разработки параллельных программ
- **Специализированные прикладные пакеты**

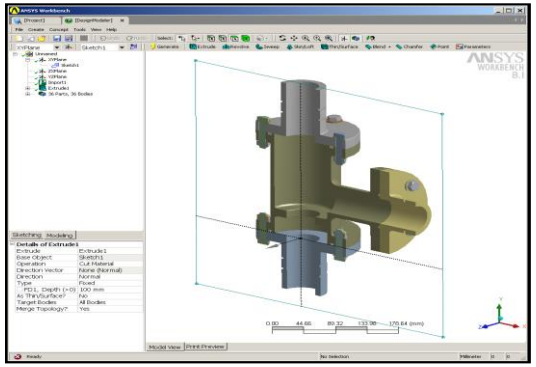
# Специализированные прикладные пакеты



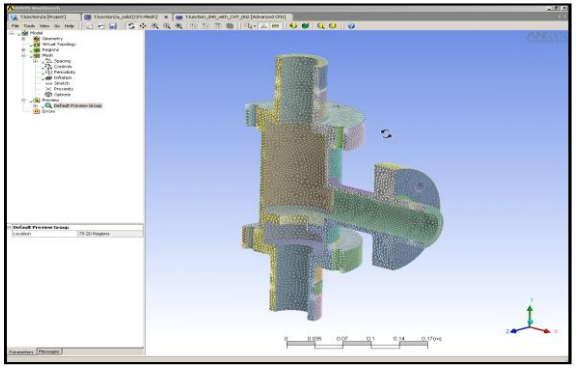
- Специализированные прикладные пакеты, работающие на параллельных вычислительных платформах
  - Задачи инженерного анализа, прочности, теплофизики, деформации, упругости, пластичности, электромагнетизма
    - **ANSYS**, MSC.NASTRAN, ABAQUS, LS-DYNA
  - Задачи аэро- и гидродинамики, механики жидкостей и газов, горения и детонации
    - CFX, FLUENT, STAR-CD, FLOWVISION, FLOW-3D, GDT
  - Задачи акустического анализа
    - LMS Virtual Lab. Acoustic, COMET/Acoustics

# Пакеты ANSYS и CFX: расчет напряжения трубы

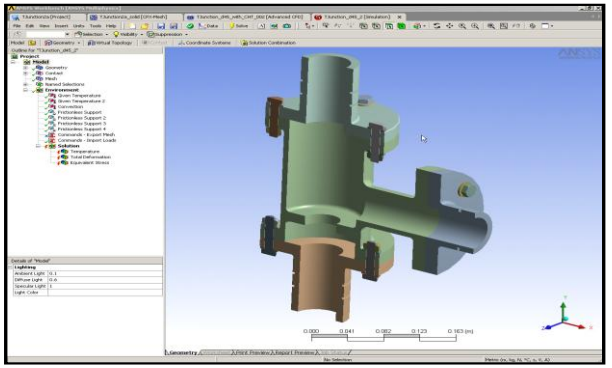




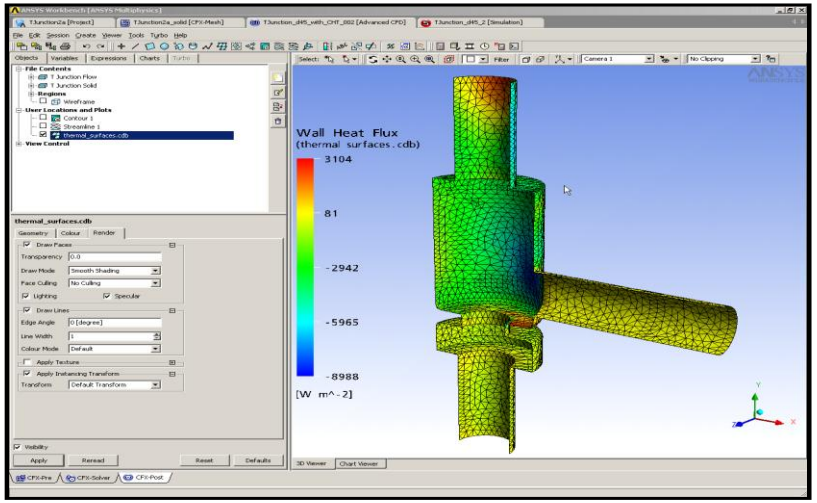
Создание геометрии модели (ANSYS)



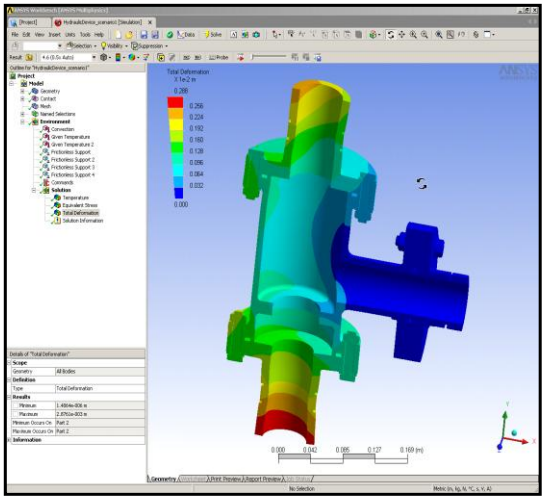
Создание сетки (ANSYS)



Настройка тепловых напряжений (ANSYS)




Расчет тепловых нагрузок (CFX)



Расчет напряжения от температуры (ANSYS)