

# Направления

- Высокопроизводительные вычисления (high-performance computation)
- Облачные технологии и технологии распределенной обработки (map-reduce)
- Обработка больших данных (data mining, big data, ..)
- Интернет вещей (IoT)

# Высокопроизводительные вычисления

- выбор вычислительной модели;
- использование эффективного вычислительного алгоритма;
- использование оптимальных структур данных и средств кодирования;
- использование оптимизированных библиотек;
- оптимизация при компиляции программы;
- оптимизация готовой программы на основе её выполнения;
- параллельное программирование;

# Внедрение параллельного программирования

- Применение компиляторов, автоматически распараллеливающих фрагменты кода
- Применение специализированных библиотек, реализующих параллельные алгоритмы вычислений
- Применение специализированных пакетов расчета
- Применение технологий параллельного программирования.

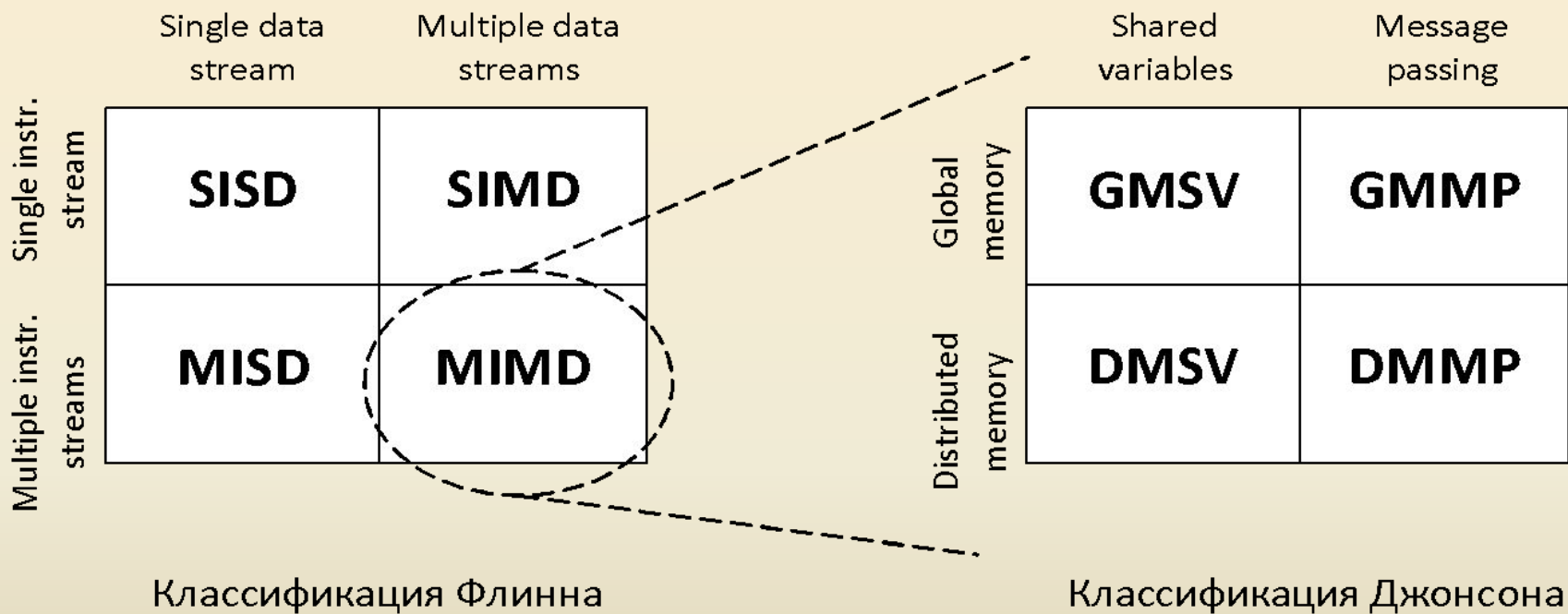
# Автоматически распараллеливающие компиляторы

- Некоторые компиляторы позволяют выполнять автоматическое распараллеливание фрагментов кода программы. Б
- Большинство распараллеливающих компиляторов работают со специализированными языками, которые упрощают задачу выделения независимых участков кода.
- Компиляторы универсальных языков высокого уровня (С/С++) позволяют распараллеливать только участки с явной независимой обработкой (например, циклическая обработка).

```
for (int i = 0; i < N; i++)  
    a[i] = b[i] + c[i];
```

```
for (int i = 0; i < N; i++)  
    x[a[i]] = x[b[i]] + x[c[i]];
```

# Классификация вычислительных систем

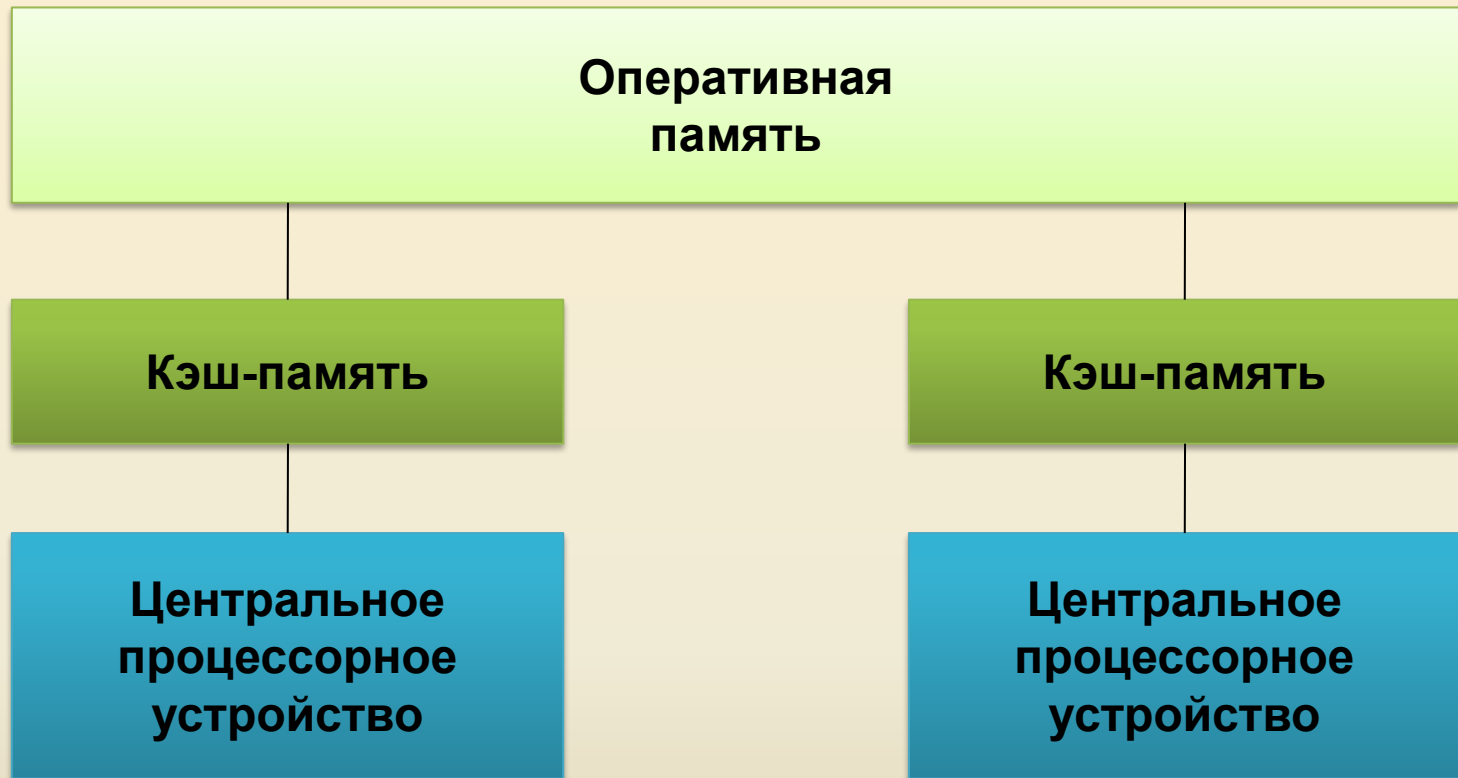


# Распределенные системы



Для взаимодействия, как правило, используются интерфейсы передачи сообщений (MPI, WCF, ..)

# Многопроцессорные системы



Системы с общей памятью (shared memory systems), многоядерные системы

# Технологии параллельного программирования

## Распределенные системы

- Стандарт MPI
- Erlang
- Go

## Многопроцессорные системы

- Стандарт OpenMP
- Task Parallel Library (C#)
- Intel Cilk Plus
- Intel TBB

## Гибридные системы

- Nvidia CUDA
- OpenACC
- OpenCL
- MS AMP C++
- MC#



# Технологии ПП для многоядерных систем

- Низкоуровневые средства (потоки)
  - Библиотеки для C/C++: pthreads, Windows Threads, boost::thread, std::thread (C++ 11);
  - Библиотеки для C#, Java, Python, ..
- Высокоуровневые средства (задачи)
  - Стандарт OpenMP (C++, Fortran)
  - Расширение Intel Cilk Plus (C++)
  - Библиотека Threading Building Blocks (C++)
  - Библиотека Parallel Primitives Library (C++)
  - Библиотека Task Parallel Library (C#)
  - ..

# Потоки vs. задачи

- Работа с потоками предполагает:
  - декомпозиция задачи на части;
  - создание и управление потоками;
  - синхронизация потоков;
  - балансировка нагрузки потоков
  - агрегирование результатов;
- Работа с задачами упрощает разработку за счет **планировщика**, который самостоятельно подбирает оптимальное число потоков, выполняет декомпозицию и агрегирование данных, динамически балансирует нагрузку

# Стандарт OpenMP

- Стандарт для многопроцессорного программирования на языках C/C++/Fortran
- Развивается с 1997 г. Последнее обновление: версия 4.5 (2015 г.)
- Поддерживается большинством компиляторов (Visual Studio, Intel C++, gcc, ..)
- Средства распараллеливания – директивы

```
// Параллельный цикл с OpenMP
#pragma omp parallel for
for (int i = 0; i < n; i++)
    y[i] = a * x[i] + y[i];
```

# Intel Cilk Plus

- Расширение для языка C++ от Intel
- Средства распараллеливания: 3 ключевых слова, расширенная векторная нотация, гиперобъекты, ..

// Параллельный цикл

```
cilk_for (int i = 0; i < n; i++)  
    y[i] = a * x[i] + y[i];
```

// Расширенная индексная нотация:

```
y[0:n] = a * x[0:n] + y[0:n];
```

# Библиотека Intel TBB

- Кроссплатформенная библиотека шаблонов C++, разработанная компанией Intel для параллельного программирования.
- Включает типовые шаблоны распараллеливания, структуры данных для многопоточных сценариев, средства синхронизации

```
tbb::parallel_for(  
    tbb::blocked_range<int>(0, n),  
    [&](tbb::blocked_range<int> r) {  
        for (int i=r.begin(); i != r.end(); ++i)  
            y[i] = a * x[i] + y[i];  
    }  
);
```

# Технологии программирования распределенных систем

- Интерфейсы передачи сообщений (MPI, WCF, ..)
- Библиотеки-оболочки над интерфейсом MPI (OO-MPI, boost mpi, MPI.NET, Global Arrays, ..)
- Языки с встроенной поддержкой обмена сообщениями (Go, Erlang, Chapel, X10, ..)
- Технологии MapReduce (apache hadoop, microsoft hdinsight, ..)

# MPI-программа

```
int rank;
float msg = 0.0;
MPI_Status status;
MPI_Init();
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
if (rank == 0)
{
    msg = 3.14;
    printf("Sending message..\n", size);
    MPI_Send(&msg, 1, MPI_FLOAT, 1, 0, MPI_COMM_WORLD);
}
else if (rank == 1)
{
    MPI_Recv(&msg, 1, MPI_FLOAT, 0, 0,
            MPI_COMM_WORLD, &status);
    printf("Received message: %f\n", msg);
}
MPI_Finalize();
```

# MPI.NET

```
static void Main(string[] args)
{
    using (new MPI.Environment(ref args))
    {
        Intracommunicator comm = Communicator.world;
        string name = MPI.Environment.ProcessorName;
        string[] names = comm.Gather(name, 0);
        if (comm.Rank == 0)
        {
            Array.Sort(names);
            foreach(string host in names)
                Console.WriteLine(host);
        }
    }
}
```

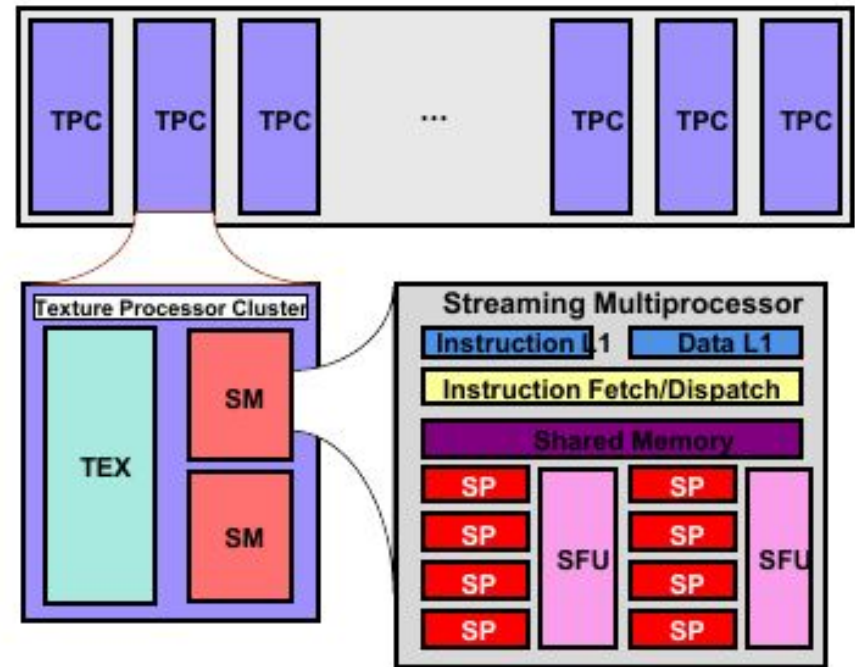


# Программирование GPU

- Современные графические процессоры (GPU) представляют собой гибко программируемые массивно-параллельные вычислительные устройства с высоким быстродействием и большим объемом собственной памяти
- Основные преимущества применения графических процессоров для параллельных вычислений связаны: с энергетической эффективностью, максимальным соотношением производительности к цене, высоким быстродействием
- Графический процессор является сопроцессором центрального процессора, обладает возможностью параллельного выполнения огромного количества отдельных нитей

# Организация GPU-устройств (G80)

- 8 ПОТОЧНЫХ процессоров (SP)
- Каждый SP-процессор исполняет 32 потока
- 2 специальных процессора SFU для сложных функций



## Средства разработки для GPU

```
graph TD; A[Средства разработки для GPU] --> B[Графические интерфейсы]; A --> C[Специализированные средства]; A --> D[Средства для гетерогенных архитектур]; B --- B1[DirectX]; B --- B2[OpenGL]; C --- C1[NVidia CUDA]; C --- C2[AMD FireStream]; D --- D1[OpenCL]; D --- D2[OpenACC]; D --- D3[MC#]; D --- D4[AMP C++]
```

### Графические интерфейсы

DirectX  
OpenGL

### Специализированные средства

NVidia CUDA  
AMD FireStream

### Средства для гетерогенных архитектур

OpenCL  
OpenACC  
MC#  
AMP C++

# Графические интерфейсы

- Данные средства были разработаны для решения задач визуализации, поэтому при их использовании для задач общего назначения появляется «привязка к графике».
- Используя графические интерфейсы (OpenGL и Direct3D), осуществляется подготовка текстур, содержащих необходимые входные данные, и через операцию *рендеринга* на графическом процессоре запускается программа для их обработки
- Недостатки данного подхода связаны с тем, что использование возможностей GPU происходит через интерфейсы, ориентированные на работу с графикой. Так в графических API полностью отсутствует возможность взаимодействия между параллельно обрабатываемыми пикселями.

# Специализированные интерфейсы

- Специализированные средства от производителей позволяют разрабатывать программу на диалекте языка C и отказаться от специфической терминологии компьютерной графики.
- Библиотеки не универсальны: они разработаны для поддержки оборудования одного производителя, и перенос программы с одной архитектуры на другую может повлечь значительные изменения кода (также это касается некоторых техник оптимизации, специфичных для одной из платформ).

# Средства гетерогенного программирования

- Данные средства реализуют модель гетерогенных вычислений, которая предполагает возможность использования в одной программе ресурсов центральных процессоров (или ядер одного процессора) и графических процессоров разных производителей
- К этой группе относятся: платформонезависимые стандарты OpenCL, OpenACC, библиотеки и расширения для высокоуровневых языков (AMP C++, MS#, Thrust Parallel Library, OpenTK).

# Интерфейс CUDA

- Технология CUDA (Compute Unified Device Architecture) представляет собой архитектуру параллельных вычислений на графических процессорах компании NVIDIA.
- Данная технология включает в себя расширения стандартного языка C (с элементами C++) для разработки параллельных приложений на графическом процессоре, набор оптимизированных библиотек (быстрое преобразование Фурье, базовые операции линейной алгебры и др.), специальный драйвер CUDA для вычислений с быстрой передачей данных между центральным и графическим процессором, а также драйвер CUDA, взаимодействующий с графическими интерфейсами OpenGL и Direct3D.

# Программирование GPU с помощью CUDA

## Общая схема исполнения программ

1. Подготовка данных на CPU
2. Копирование данных в память GPU
3. Выполнение вычислительных функций (ядер) на GPU
4. Копирование результатов работы в память CPU.

// Определение функции-ядра

```
__global__ void matAdd ( float * A, float * B, float * C, int N) {  
    int i = blockIdx.x * blockDim.x + threadIdx.x;  
    int j = blockIdx.y * blockDim.y + threadIdx.y;  
    if ( i < N && j < N )  
        C [i * N + j] = A [i * N + j] + B [i * N + j];  
}
```



# “Hello-world” на CUDA

```
// Выделяем память на GPU для копий переменных a,  
b, c  
cudaMalloc((void **)&d_a, size);  
cudaMalloc((void **)&d_b, size);  
cudaMalloc((void **)&d_c, size);  
// Копируем данные на устройство  
cudaMemcpy(d_a, &a, size, cudaMemcpyHostToDevice);  
cudaMemcpy(d_b, &b, size, cudaMemcpyHostToDevice);  
// Запускаем функцию ядра на графическом  
процессоре  
add<<<N,1>>>(d_a, d_b, d_c);  
// Сохраняем результаты на хосте  
cudaMemcpy(&c, d_c, size, cudaMemcpyDeviceToHost);  
// Очищаем память  
cudaFree(d_a); cudaFree(d_b); cudaFree(d_c);
```

# Стандарт OpenCL

- OpenCL (Open Computing Language) является открытым межплатформенным стандартом для параллельных вычислений на современных процессорах разных типов.
- Основное внимание в стандарте сосредоточено на поддержке многоядерных центральных процессоров и графических ускорителей.
- Стандарт предоставляет программистам переносимый и эффективный доступ ко всей мощности гетерогенных вычислительных платформ.

# Стандарт OpenACC

- Стандарт для упрощения программирования GPU разных производителей

## **“OpenACC: More Science Less Programming”**

- Поддерживается компиляторами: PGI Accelerator Compiler, Cray Compilation Environment, PathScale Enzo Compiler. Ограниченная поддержка в GCC 6.0
- Основные директивы: `#pragma acc parallel`, `#pragma acc kernels`, `#pragma acc data`, `#pragma acc loop`, `#pragma acc cache`, `#pragma acc update`, `#pragma acc declare`, `#pragma acc wait`

# Пример OpenACC-функции

```
// Умножение вектора на число
void VectorMultiplication(int n, float a, float *x, float * y) {
    #pragma acc kernels
    for (int i = 0; i < n; ++i)
        y[i] = a * x[i];
}

// Вычисление числа Пи
double CalcPi(long N) {
    double pi = 0.0f; long i;
    #pragma acc parallel loop reduction(+:pi)
    for (i=0; i<N; i++) {
        double t= (double)((i+0.5)/N);
        pi +=4.0/(1.0+t*t);
    }
    return pi;
}
```

# Вычисления на GPU на других языках

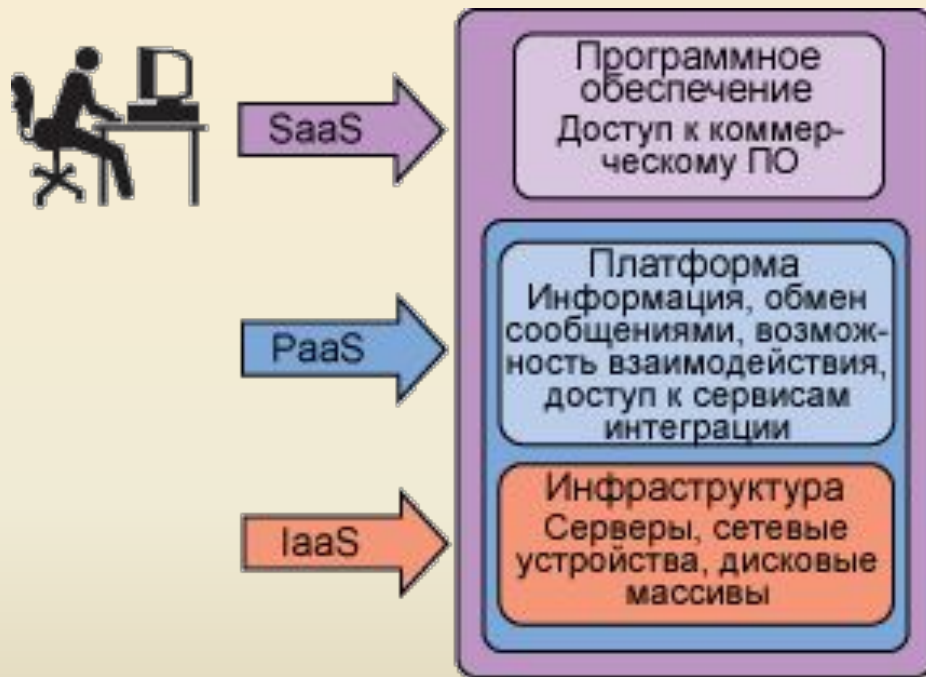
- Для языка C#: Cloo, OpenCL.NET, CUDA.NET
- Для языка F#: AleaGPU, Brahma
- Для языка Python: Anaconda Accelerate, PyCuda,
- ..

<https://developer.nvidia.com/language-solutions>

# Облачные вычисления

- Комплексное решение, предоставляющее компьютерные ресурсы в виде сервиса.
- Ресурсы могут предоставляться через Интернет или локальную сеть и доступны на различных платформах и устройствах.
- В качестве основных ресурсов выступают: вычислительные мощности (аппаратные ресурсы), программное обеспечение, данные.
- Компьютеры в облаке настроены на совместную работу, а различные приложения используют совокупную вычислительную мощность так, как будто выполняются на одиночной системе.

# Уровни облачных вычислений



В облачной инфраструктуре выделяют несколько уровней, каждый из которых обеспечивает тот или иной вид сервиса для потребителей: уровень инфраструктуры, уровень платформы и уровень приложений.

# Уровень инфраструктуры

- Уровень инфраструктуры или уровень IaaS (*Infrastructure as a Service* — *инфраструктура как сервис*) предоставляет аппаратные средства и системное программное обеспечение в качестве ресурса Ресурсы: серверы, системы хранения данных, клиентские системы, сетевое оборудование. Системное программное обеспечение включает операционные системы, средства виртуализации, автоматизации.
- Использование технологий уровня IaaS избавляет клиента (предприятие) от необходимости поддержки сложных инфраструктур центров обработки данных, клиентских и сетевых инфраструктур. Как правило, пользователь сервисов инфраструктуры оплачивает реально используемые ресурсы: серверное время, дисковое пространство, сетевую пропускную способность и т. д.
- Примерами облачных сервисов, предоставляющих инфраструктурные услуги: Elastic Compute Cloud (EC2), Simple Storage Service (S3), GoGrid, Enomaly, Eucalyptus. Сервис EC2 позволяет арендовать образы виртуальных машин. Сервис S3 предназначен для хранения данных.



# Уровень платформы

- Уровень платформы или уровень PaaS (*Platform as a Service* — платформа как сервис) — это предоставление интегрированной платформы для разработки, тестирования, развертывания и поддержки приложений.
- Для разворачивания приложений разработчику не нужно приобретать оборудование и программное обеспечение, нет необходимости организовывать их поддержку.
- Существующие технологии уровня PaaS, как правило, позволяют использовать определенные языки разработки. Примерами облачных технологий, интегрирующих среду для разработки, являются: платформа Google AppEngine (разработка на языках Java, Python, Go), Engine Yard (Ruby), PHP Fog, Stackato (Perl), Cloudbees (Java), Windows Azure (.NET, Java, PHP, Ruby), OpenShift (Java, Ruby, PHP, Python).

# Уровень программного обеспечения

- Уровень приложений или уровень SaaS (*Software as a Service — программное обеспечение как сервис*) — модель развертывания приложения, которая подразумевает предоставление приложения конечному пользователю как услуги по требованию.
- Приложение приспособлено для удаленного использования, одним приложением могут пользоваться несколько клиентов.
- Наибольшим спросом среди SaaS-приложений пользуются: почта, коммуникации, антивирусные программы, программы управления проектами, дистанционное обучение, CRM, хранение и резервирование данных. Распространенными SaaS-сервисами являются: Gmail, Office 365, Google Apps, Salesforce.com.

# Типы развертывания

- *Частное облако (private cloud)* — используется для предоставления сервисов внутри одной компании, которая является одновременно и заказчиком и поставщиком услуг.
- *Публичное облако* — используется облачными провайдерами для предоставления сервисов внешним заказчикам.
- *Смешанное (гибридное) облако* — это комбинация открытого и закрытого облака.

# Достоинства и недостатки облачных решений

- Среди основных преимуществ технологий облачных вычислений можно выделить: доступность и открытость, экономичность, простота использования, гибкость и масштабируемость.
- Среди недостатков облачных технологий можно выделить: необходимость постоянного соединения с сетью, возможность проблем с безопасностью данных, ограниченная функциональность облачных приложений, зависимость от компании, предоставляющей сервисы облачной инфраструктуры.

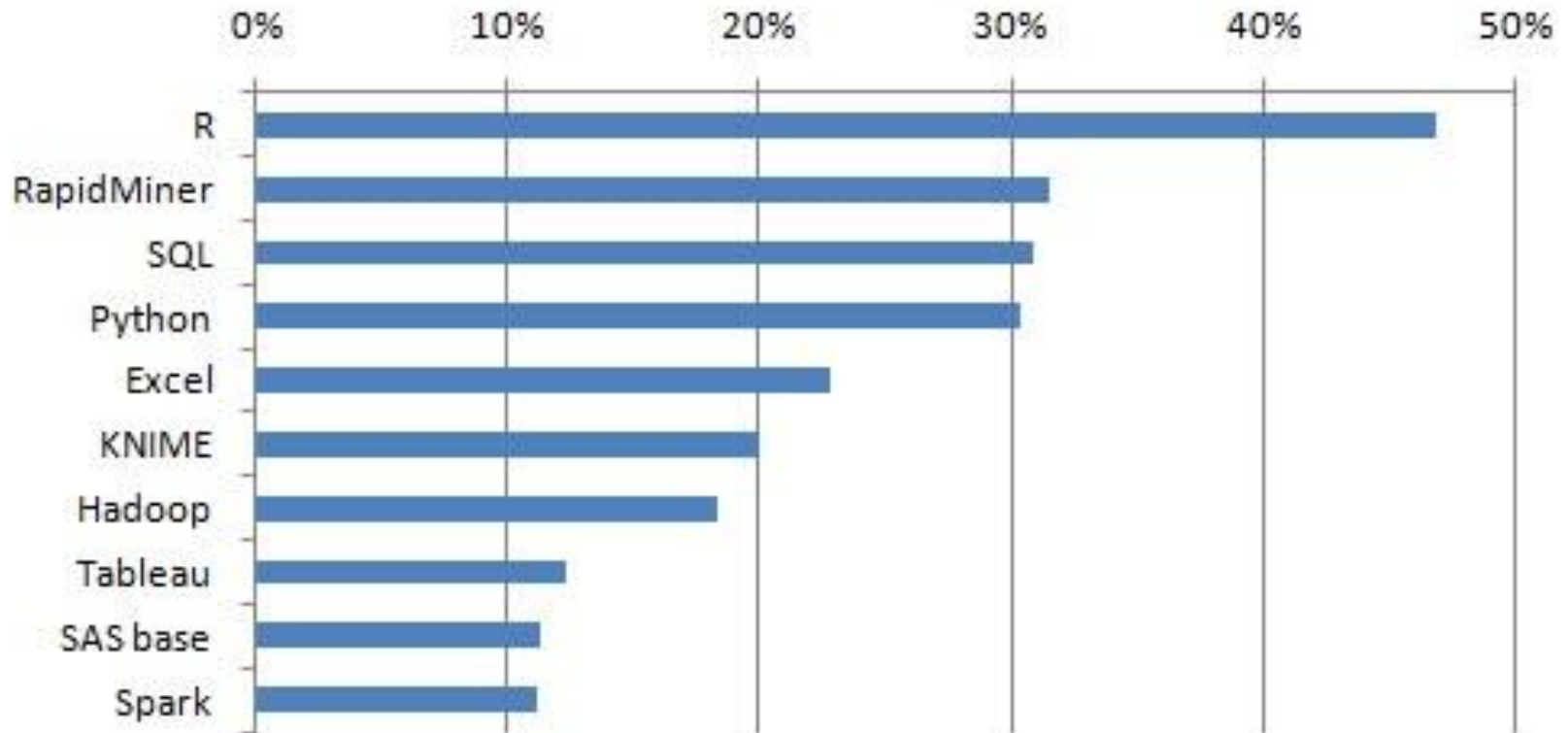
# Анализ данных

- **Data mining** - процесс извлечения закономерностей (шаблонов) в больших массивах информации.
- “DM - процесс обнаружения в сырых данных ранее неизвестных, нетривиальных, практически полезных, доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности”
- Варианты перевода: анализ данных (АД), обнаружение знание (knowledge discovery), интеллектуальный анализ данных, раскопка данных, добыча данных, ..
- Связанные и смежные области: машинное обучение (machine learning), big data, data science, ..

# Программные продукты анализа данных

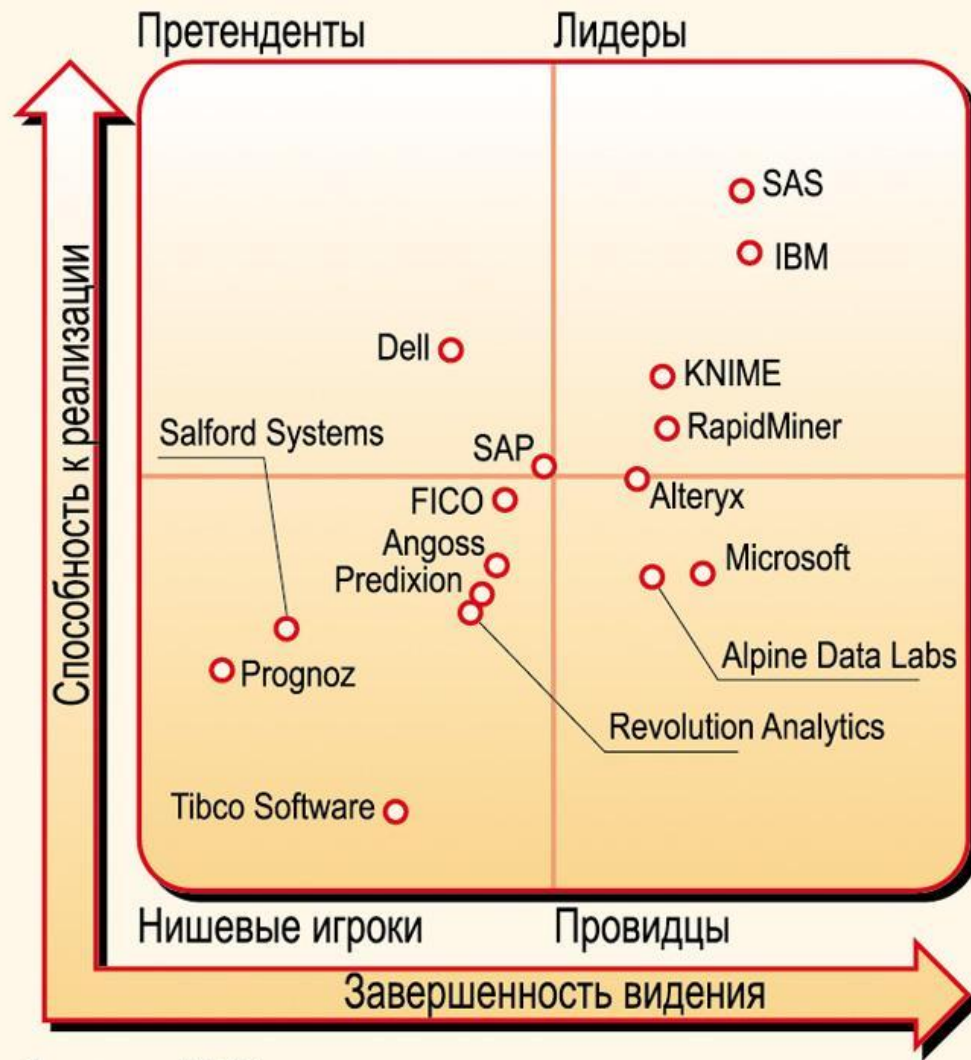
| Категория                      | Свободное                                     | Платное  |
|--------------------------------|---|--|
| Статистические пакеты          |   | StatSoft,<br>STATISTIKA,<br>S-Plus                         |
| Специализированные инструменты | GATree  | CART,<br>Viscovery SOMine,<br>WhizWhy                      |
| Универсальные пакеты           | KNIME*, Orange,<br>KEEL, Weka,<br>RapidMiner* | IBM, Oracle,<br>Microsoft SQL,<br>Deductor,<br>PolyAnalyst |
| Языки разработки               | R, Python+                                    | Xelopes  |

## Top Analytics, Data Mining, Data Science software used, 2015



<http://www.kdnuggets.com/polls/2015/analytics-data-mining-data-science-software-used.html>

# Магические квадранты для продвинутой аналитики, февраль 2015



Февраль 2015 г.

Источник: Gartner.