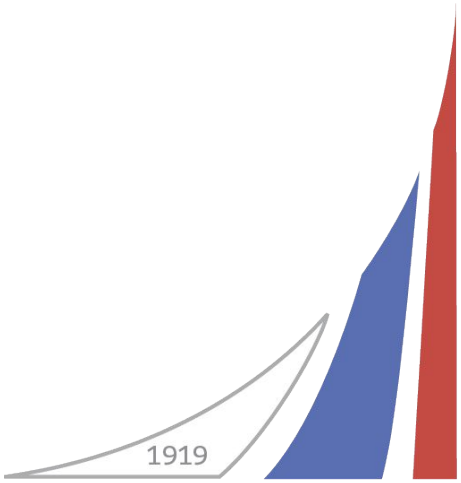


ФИНАНСОВЫЙ УНИВЕРСИТЕТ

ПРИ ПРАВИТЕЛЬСТВЕ РОССИЙСКОЙ ФЕДЕРАЦИИ

Введение в специальность

Введение в технологии параллельного программирования

Корчагин Сергей Алексеевич, кандидат физико-математических наук,
заместитель руководителя по научной работе,
доцент Департамента анализа данных и машинного обучения
SAKorchagin@fa.ru

Москва, 2021

https://regnum.ru/pictures/2255436/1.html

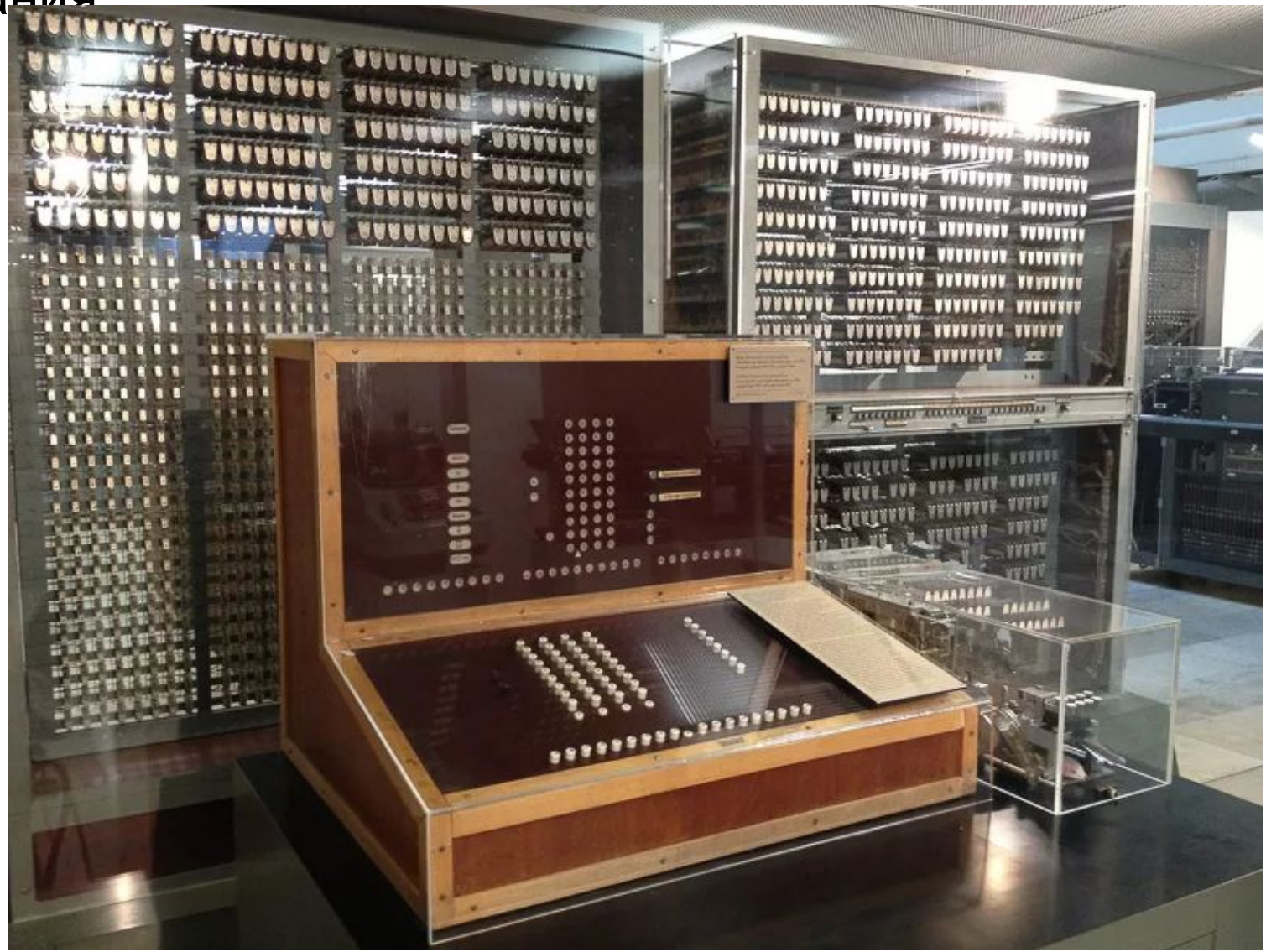
Введение в технологии параллельного программирования

- Понятие «Технологии параллельного программирования», история
- Примеры и основные области применения «Технологий параллельного программирования».
- Основные архитектурные особенности построения параллельной вычислительной среды.
- Основные классы современных параллельных компьютеров.

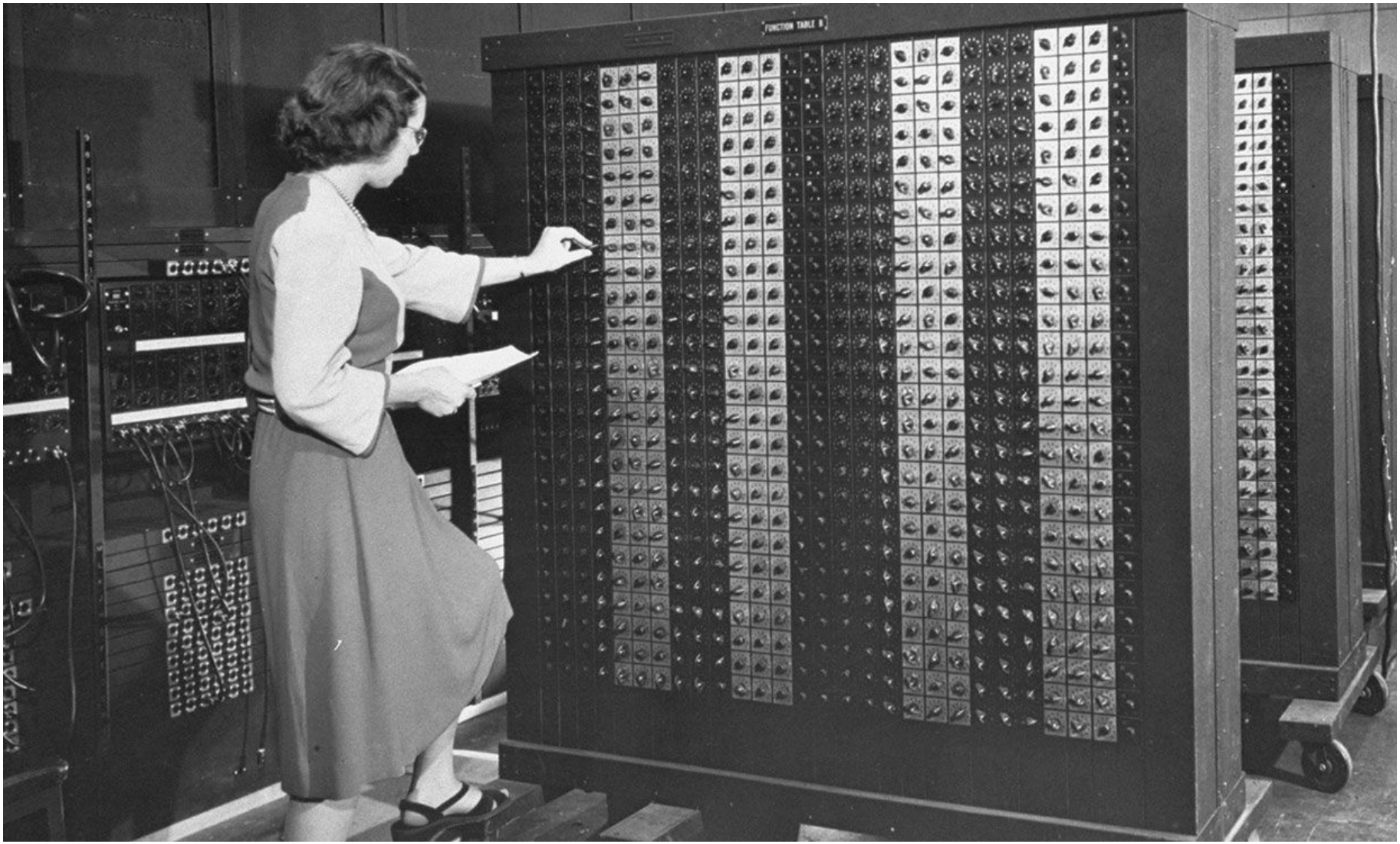
Параллельное программирование - это техника программирования, которая использует преимущества многоядерных или многопроцессорных компьютеров.

Параллельное программирование – раздел программирования, связанный с изучением и разработкой методов и средств для:
а) адекватного описания в программах естественного параллелизма моделируемых в ЭВМ и управляемых ЭВМ систем и процессов,
б) распараллеливания обработки информации в многопроцессорных и мультипрограммных ЭВМ с целью ускорения вычислений и эффективного использования ресурсов ЭВМ.

1941 г. - Конрад Цузе, вычислительная машина Z3, Германия



1945 г. - Джон Мокли, ЭВМ ЭНИАК, США



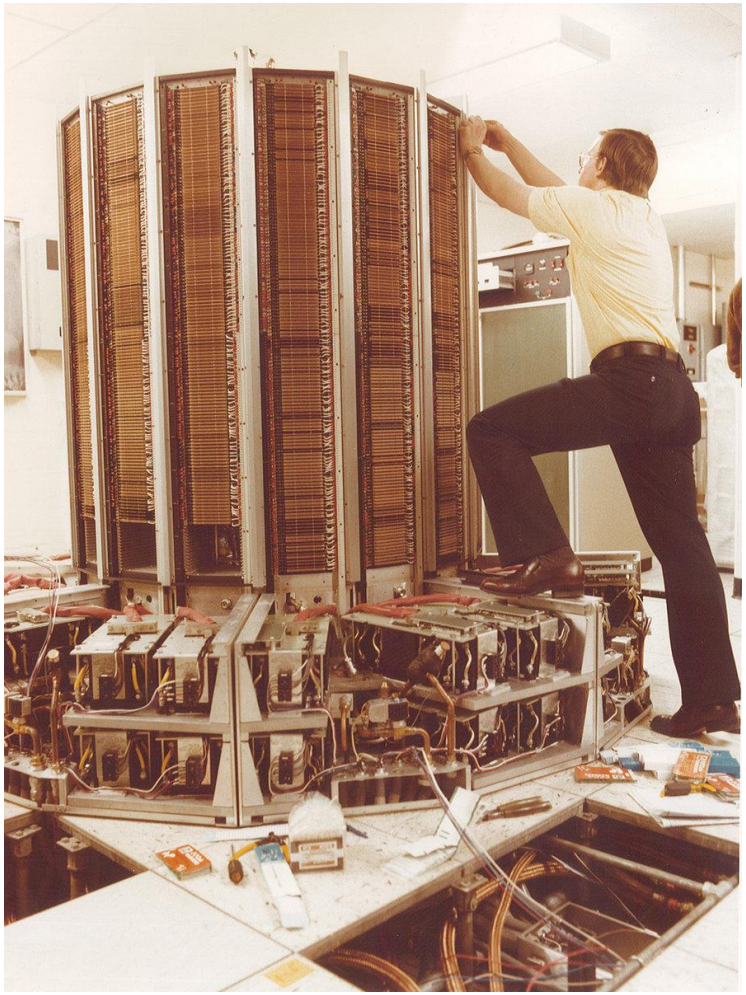
1959 г. - Анатолий Иванович Китов, ЭВМ «М-100», **СССР**

1961 г. - IBM 7030, США

1962 г. – Atlas, Манчестерский университет, Великобритания

1964 г., компания **Control Data Corporation, Сеймур Крэй, CDC-6600, США**

1976 г., компания Cray Recearch, Сеймур Крэй, **CRAY 1, США**



1982 г., компания Cray Recearch, Сеймур Крэй, **CRAY X-MP, США**

1996 г., компания Intel, Sandia NL, ASCI Red, США

ASCI Red:
Sandia National Laboratories

Number 1 system from June 1997 to June 2000

intel

Sandia National Laboratories

Intel ASCI Red Sandia National Laboratories, USA			
Date	Cores	Linpack Peak	Theoretical Peak
6/97	7,264	1.068 Tflop/s	1.453 Tflop/s
11/97	9,152	1.34 Tflop/s	1.83 Tflop/s
6/98	9,152	1.34 Tflop/s	1.83 Tflop/s
11/98	9,152	1.34 Tflop/s	1.83 Tflop/s
6/99	9,472	2.1 Tflop/s	3.1 Tflop/s
11/99	9,632	2.4 Tflop/s	3.2 Tflop/s
06/00	9,632	2.4 Tflop/s	3.2 Tflop/s
Interconnect: Proprietary Operating System: Paragon OS			

Last appearance on list: No. 276 in November 2005

2002 г., компания NEC, **Earth Simulator, Япония**

2009 г., IBM, Roadrunner, США

2020 г., Fujitsu Limited, Fugaku, Япония

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Supercomputer Fugaku** - Supercomputer Fugaku, A64FX 48C 2.2GHz, Tofu interconnect D, Fujitsu RIKEN Center for Computational Science Japan	7,299,072	415,530.0	513,854.7	28,335
2	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	**Sierra** - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438
4	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway, NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
5	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000, NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482

Рейтинг суперкомпьютеров

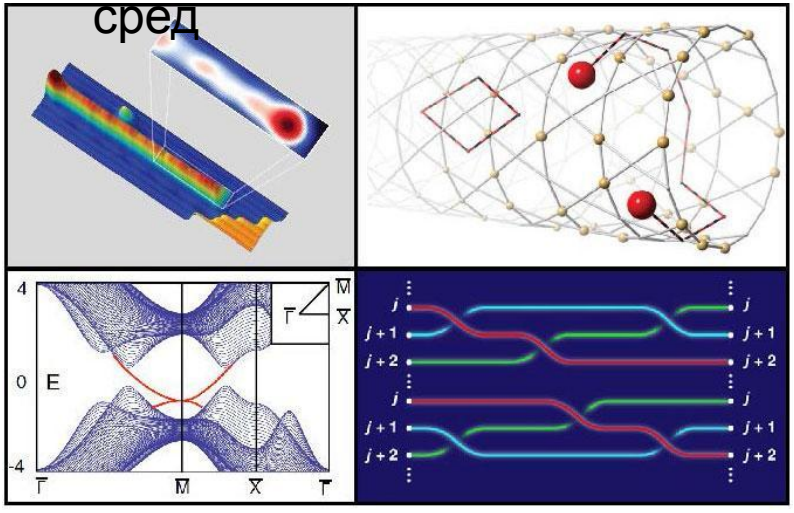
Рейтинг 32 редакции от 31 марта 2020 года[8]						
№	Местонахождение	Наименование	Год создания	CPU / ядер	Архитектура	TFlops (Linpack/ Пиковая)
1	SberCloud (ООО «Облачные технологии»), СберБанк, Москва	«Кристофари»	2019	1200 / 99 600	CPU: 2x Intel Xeon Platinum 8168 24C 2.7GHz, 1536 GB RAM Acc: 16x NVIDIA Tesla V100	6669/ 8789
2	Москва, МГУ	Ломоносов-2	2014	1472 / 42688	A-class[9][10][11]: Xeon E5-2697v3, Tesla K40M, 2.6 GHz; сети Infiniband QDR/Gigabit Ethernet	2478/ 4946
3	Москва, ФГБУ 'ГВЦ Росгидромета'		2018	1952 / ?	CPU: 2x Intel Xeon E5-2697v4, 128 GB RAM	1200 / 1293
4	Москва, МГУ	Ломоносов	2012	12422 / 82468	Xeon 5570, Tesla X2070, PowerXCell 8i; сети Infiniband QDR/Gigabit Ethernet	901 / 1700
5	Санкт-Петербург	ФГАОУ ВО СПбПУ	2014	1528 / 19936	2 x Xeon E5-2697v3 2.6 GHz; сети: FDR Infiniband/Gigabit Ethernet	801 / 1148
6	Москва	Курчатовский институт[12]	2015	296 / 10064	2x Xeon E5-2650v2 2.6 GHz; сети: FDR Infiniband/Gigabit Ethernet	755/ 1100
7	Москва	Высшая школа экономики	2015	68/	2x Intel Xeon Gold 6152, 768 GB RAM Acc: 4x NVIDIA Tesla V100	568/ 912
8	Сколково, Сколковский институт науки и технологий	«ZHORES CDISE Cluster»	2018	172 /2296	2x Intel Xeon Gold 6136; сети: FDR Infiniband/Gigabit Ethernet /Fast Ethernet	495/ 1011
9	Москва, Тинькофф банк	Колмогоров	2019	20 / 360	2x Intel Xeon Gold 6154; сети:100 Gigabit Ethernet	418/ 658
10	Москва	ФГБУН МСЦ РАН	2016	416 / 28704	Xeon E5, Xeon Phi 7110X; сети: FDR Infiniband/Gigabit Ethernet	383 / 523

http://top50.supercomputers.ru/list

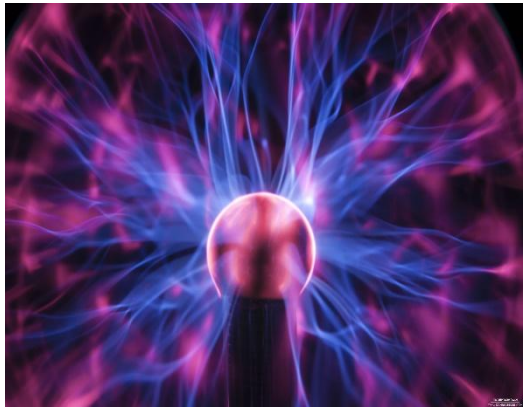
Примеры и основные области применения технологий параллельного программирования

Научные исследования в области естественных наук

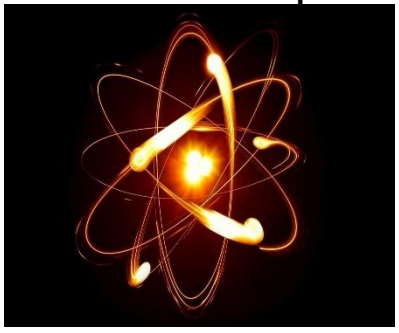
- Физика конденсированных сред



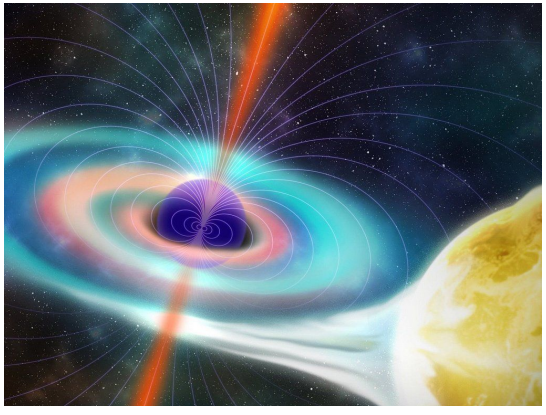
- Физика плазмы



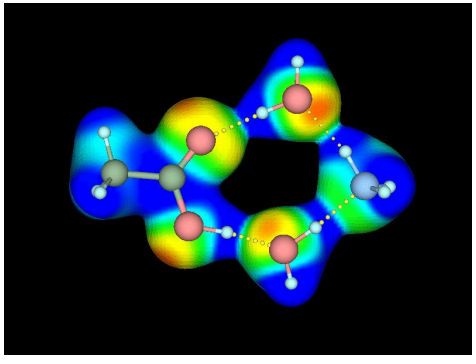
- Атомная физика



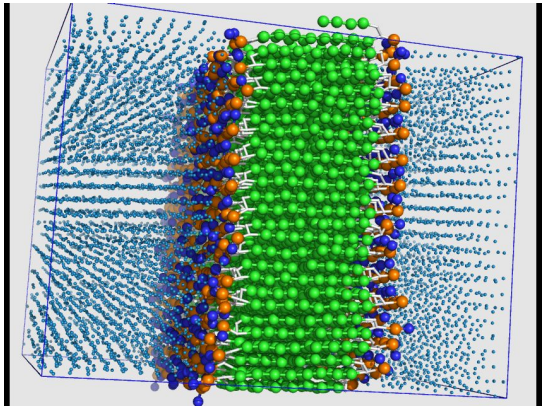
- Астрофизика



- Квантовая химия

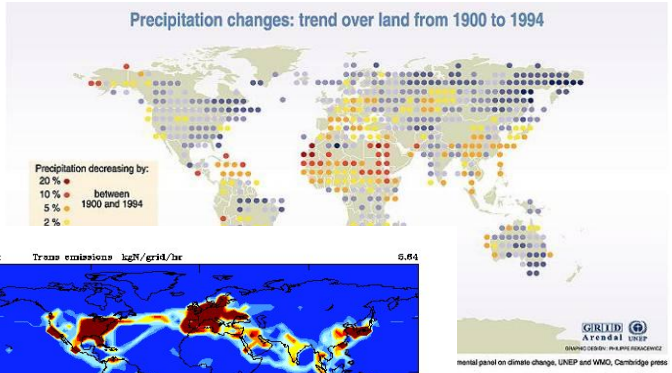


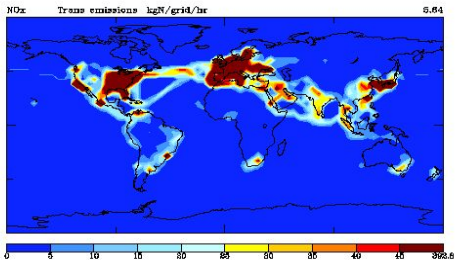
- Молекулярная динамика

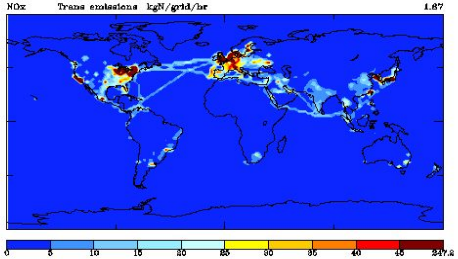


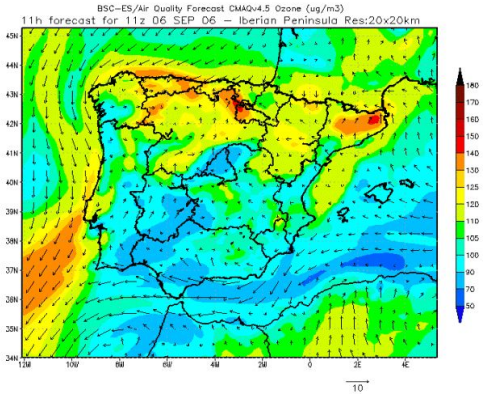
Примеры приложений: **Науки о Земле**

- Анализ изменений климата

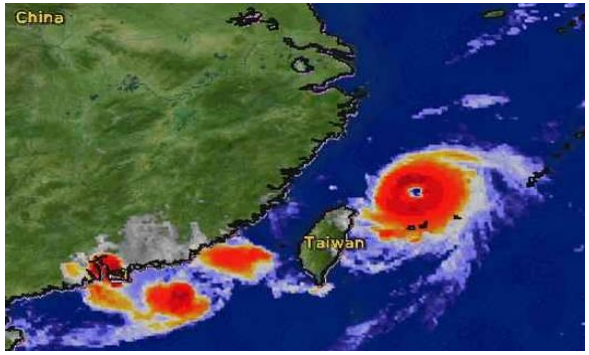




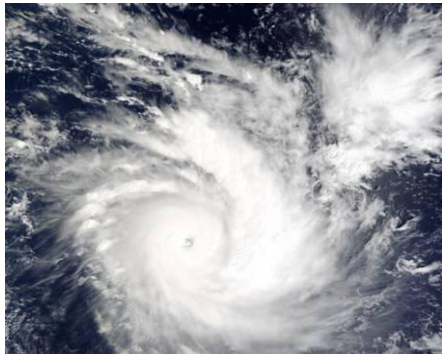


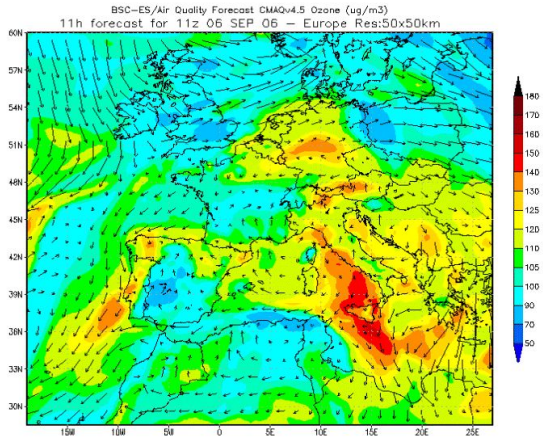


- Состояние атмосферы



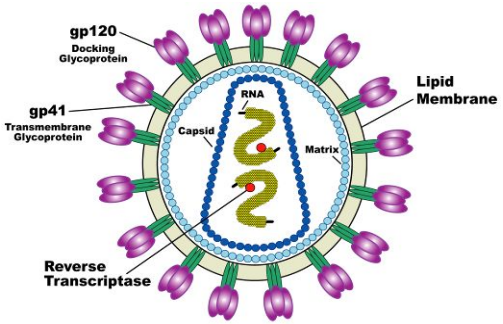
- Прогнозирование погоды

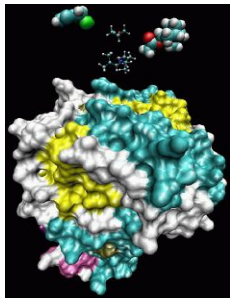




Примеры приложений: **Науки о жизни**

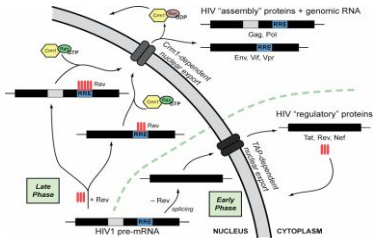
- Новые лекарства и методы лечения

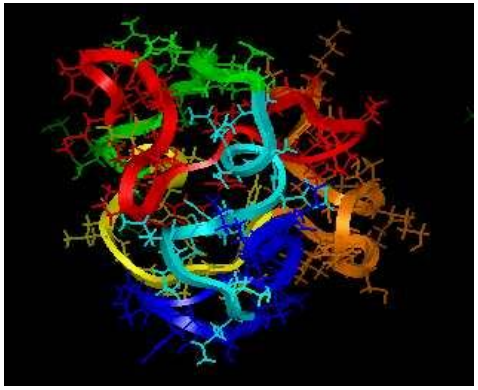


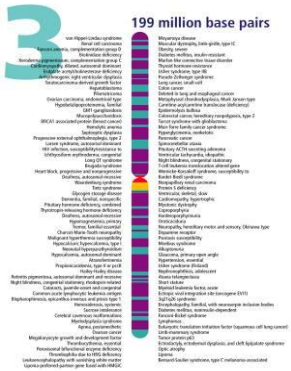


- Геномика

- Поиск в базах данных

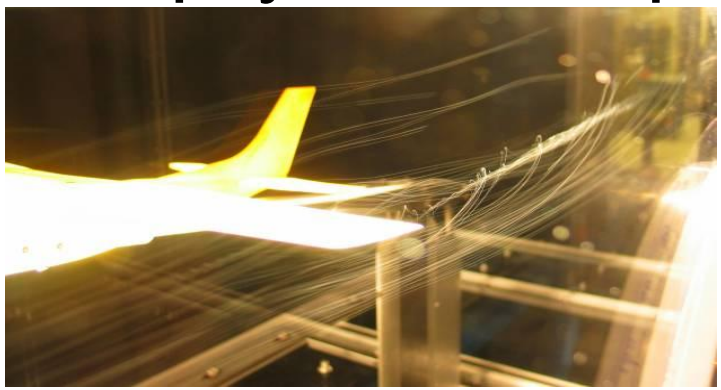


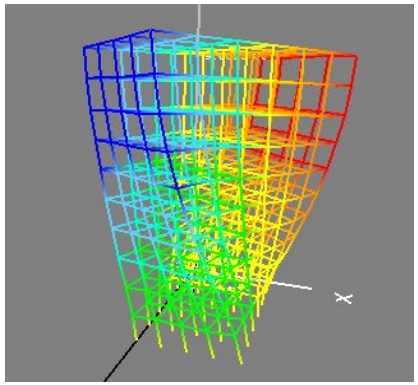


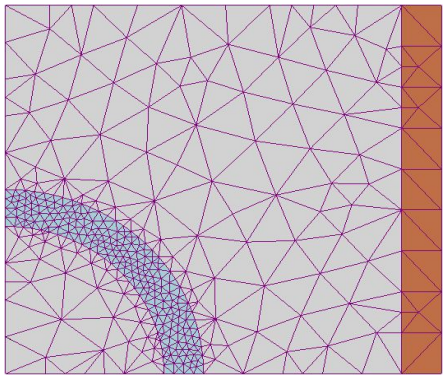


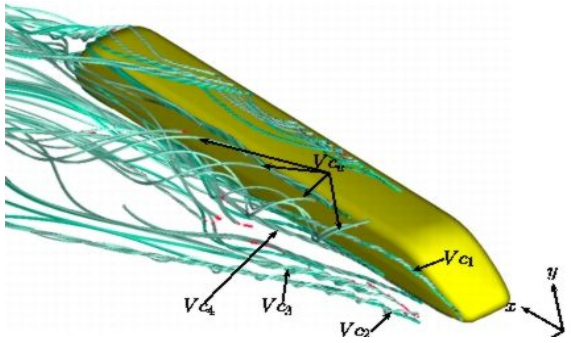
Инженерные расчёты

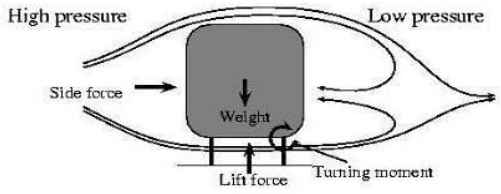
- Виртуальное проектирование
- Оптимизация

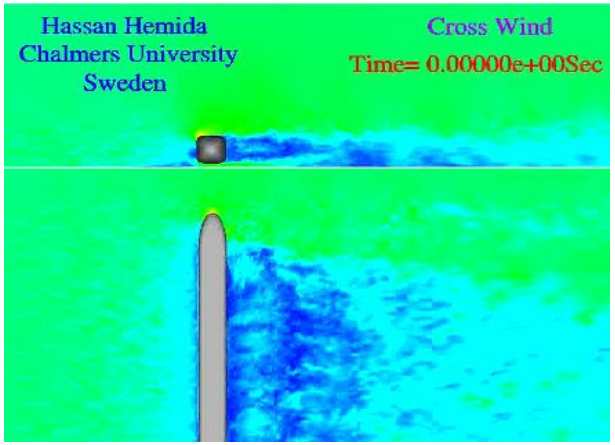






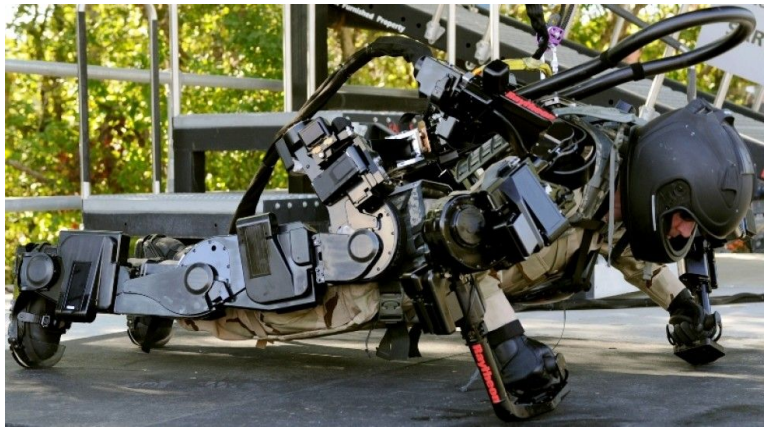






ВПК

- Проектирование экзосклетов
- Роботов



- Расшифровка информации

- Обработка снимков

- Модернизация и разработка техники

Финансовый сектор

- Сервисы на основе ИИ

- Автоматизированное принятие решений

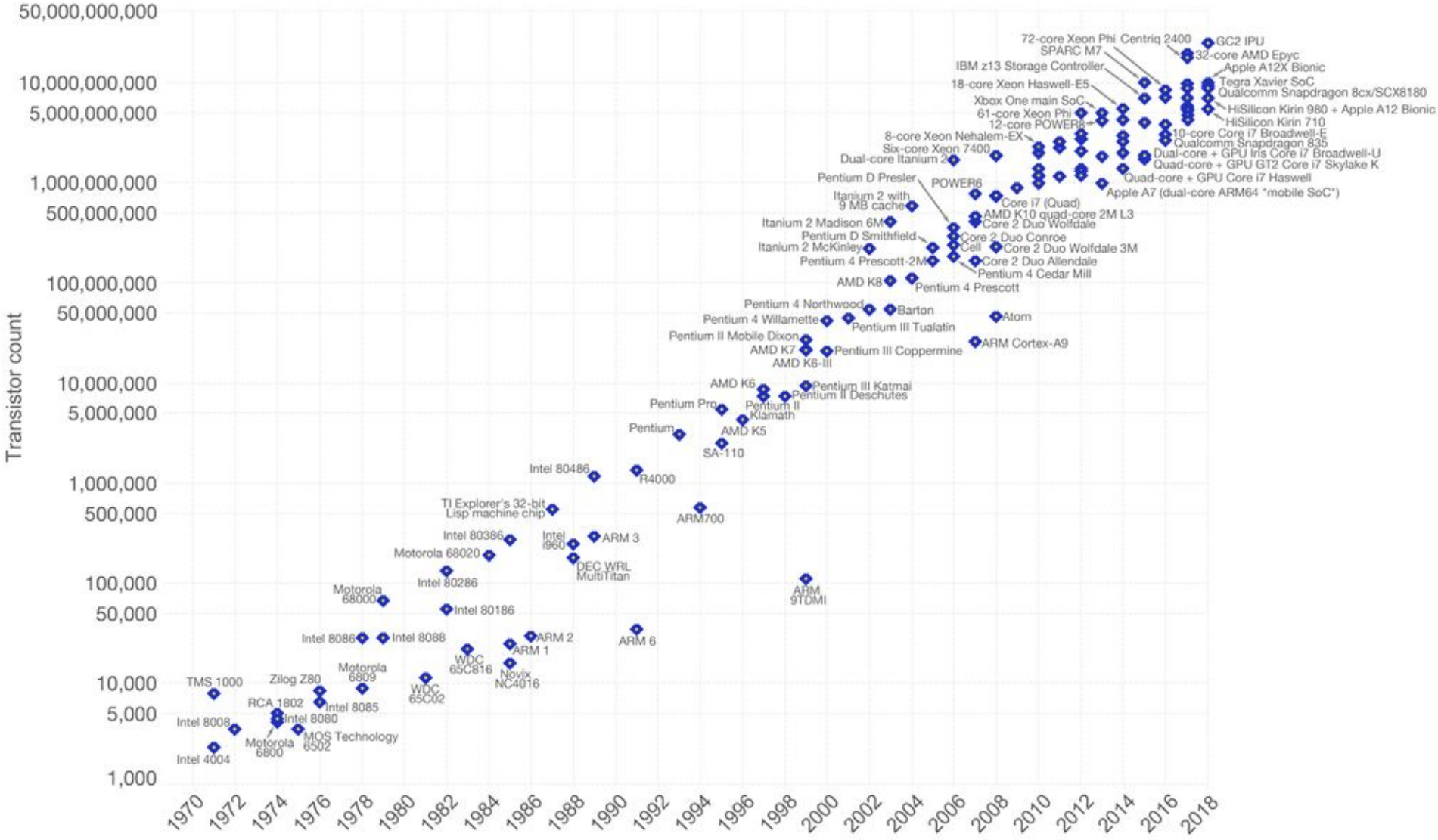
- Оценка и управление рисками

- Предиктивная аналитика

Проблемы технологий параллельного программирования

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

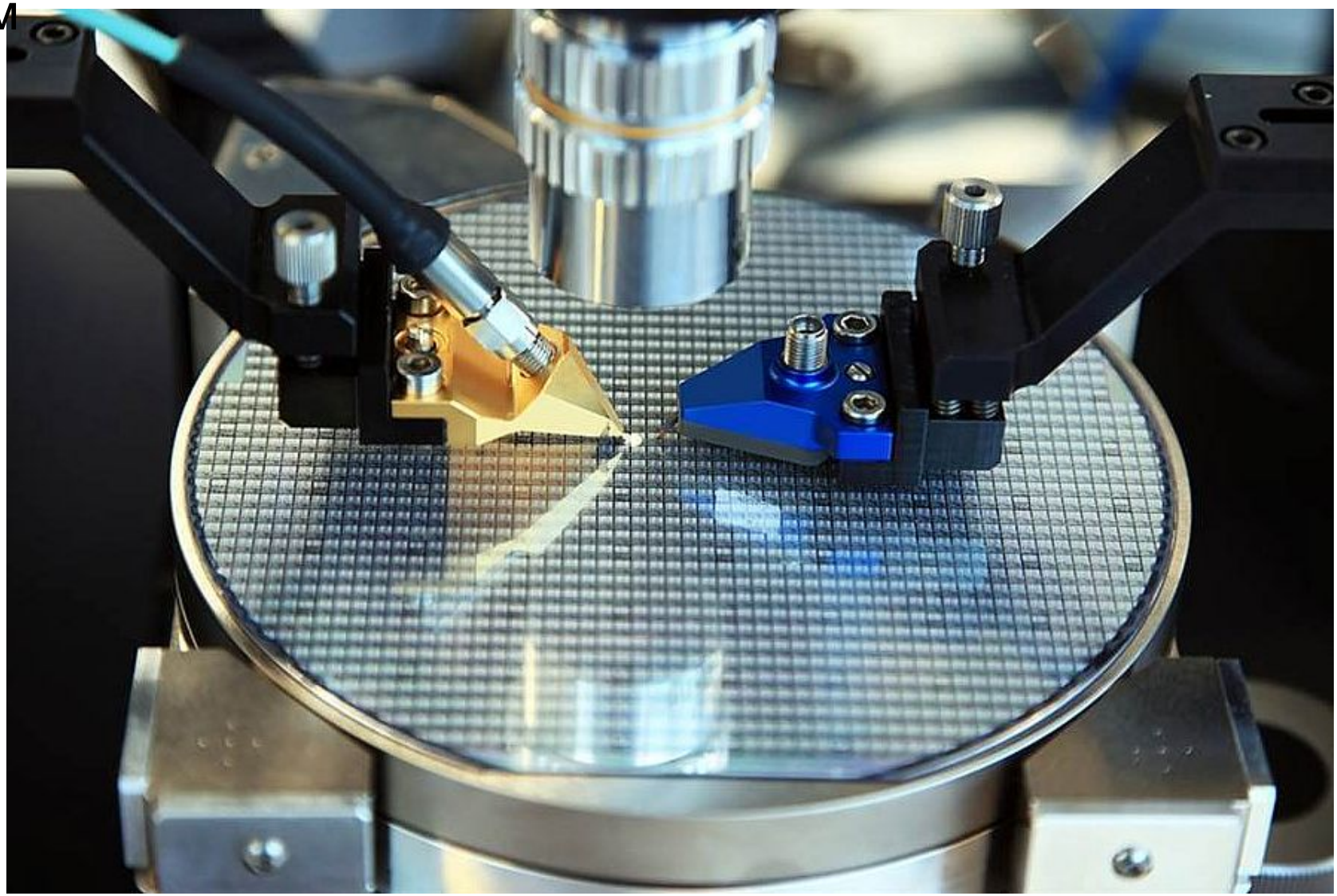
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

http://www.invertedalchemy.com/2017/05/

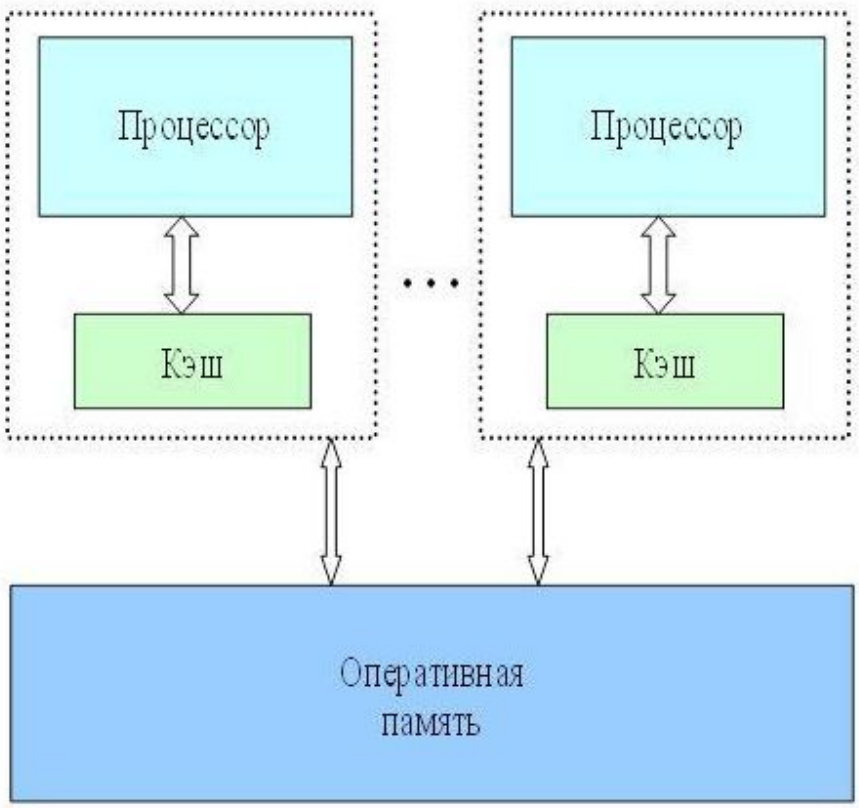
TSMC – техпроцессор - 5-нм

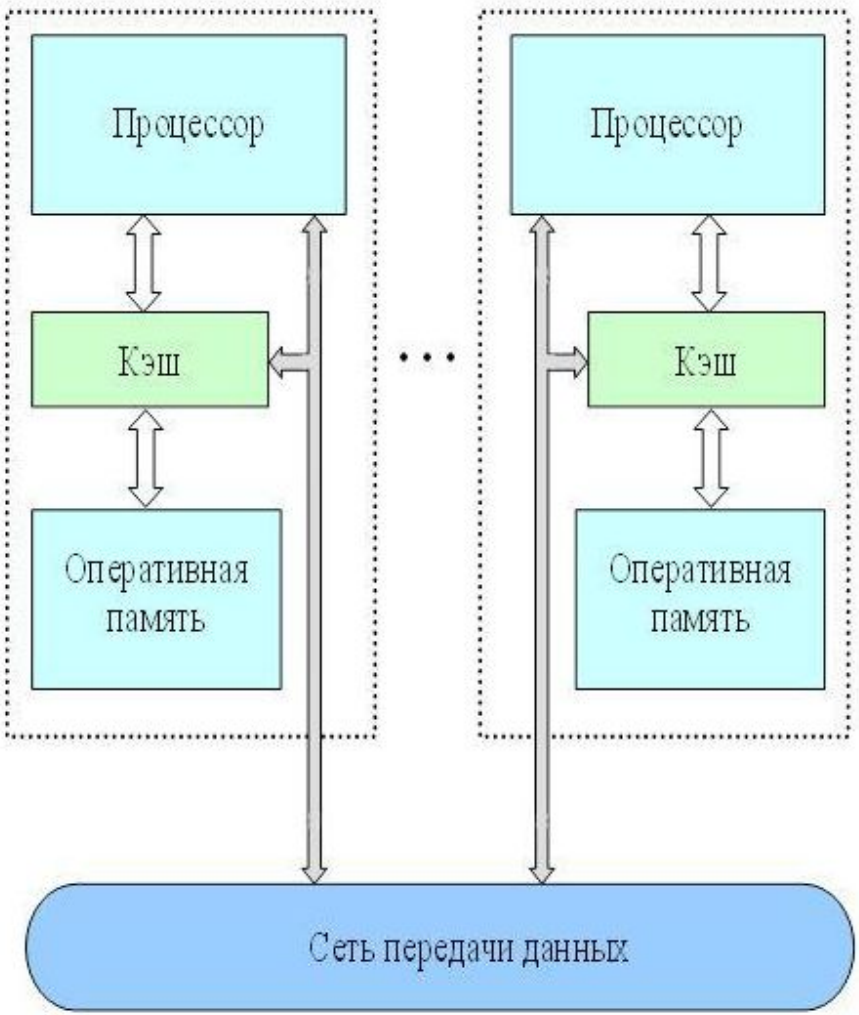


Основные архитектурные особенности построения параллельной вычислительной среды

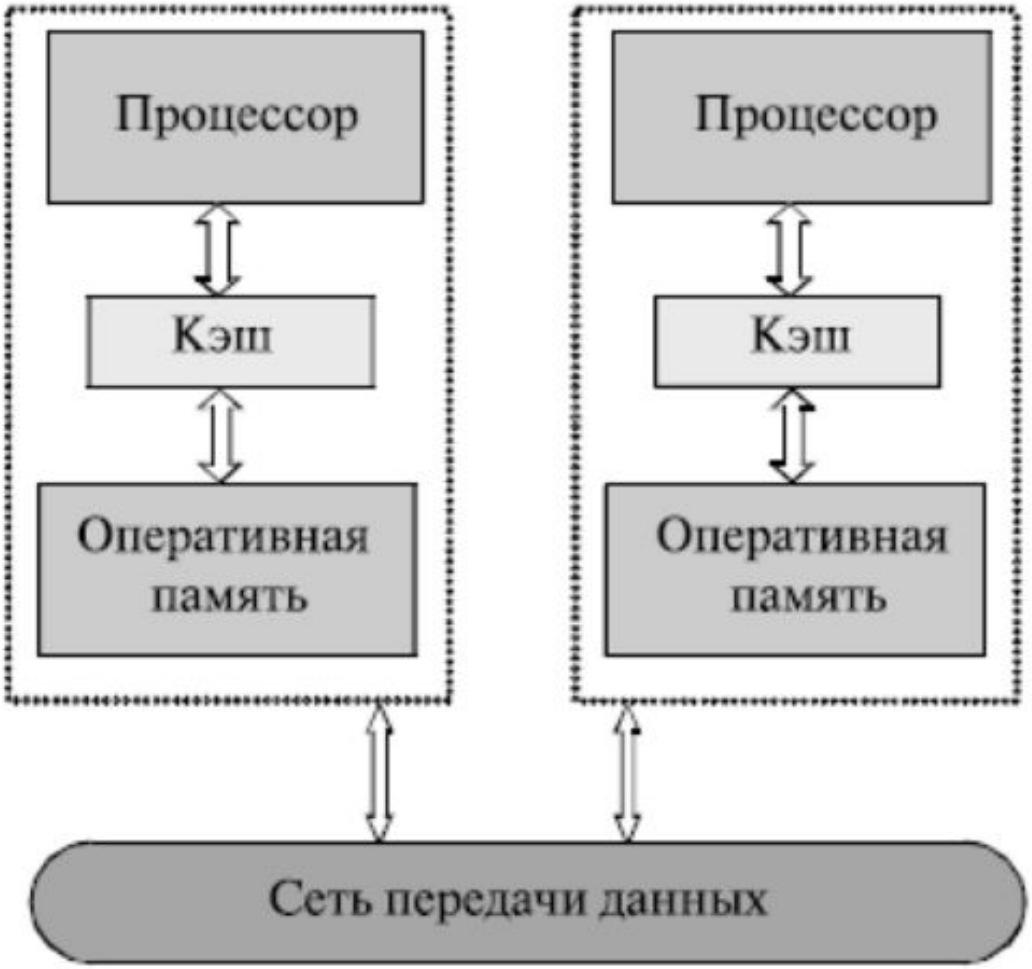
https://musicseasons.org/wp-content/uploads/MG_6427.jpg

Мультипроцессоры





Мультикомпьютеры



https://parallel.ru/computers/taxonomy/

ПЛЮСЫ И МИНУСЫ РАЗЛИЧНЫХ АРХИТЕКТУР

Архитектура с разделяемой памятью

Архитектура с распределенной памятью

ПРЕМУЩЕСТВА

- Привычная модель программирования за счет единого адресного пространства
- Высокая скорость и низкая латентность обмена данными между параллельными задачами

- Высокая масштабируемость
- Объем памяти растет пропорционально количеству ядер
- Возможность использовать недорогие массовые компоненты

НЕДОСТАТКИ

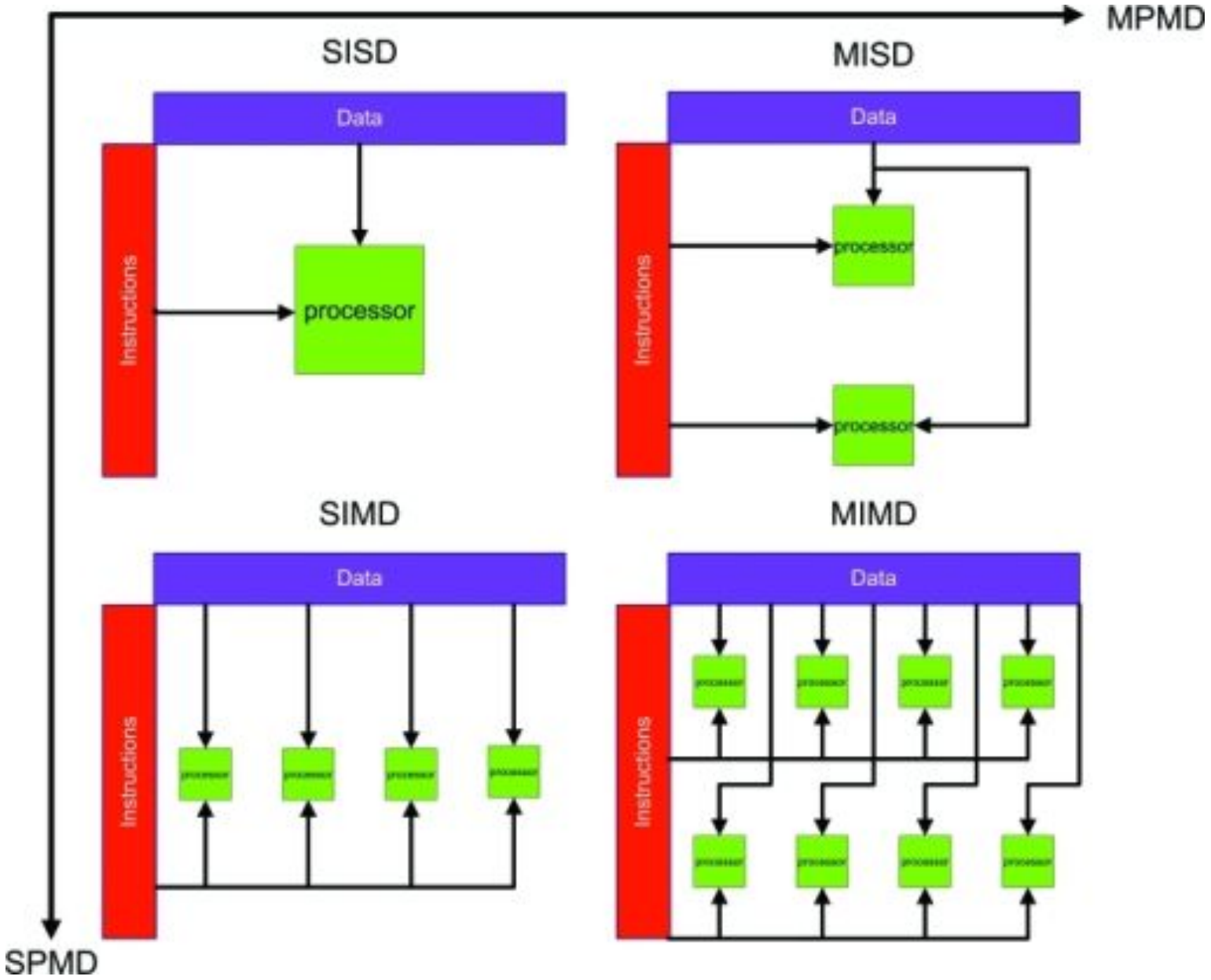
- Низкая масштабируемость (обычно до 16 процессоров) из-за геометрического роста нагрузки на шину CPU-RAM
- Проблема поддержания когерентности кэшей
- Трудоемкая организация эффективного использование памяти в NUMA-системах
- Необходимость синхронизации при доступе к общим данным (критические секции)

- Специальные подходы к программированию: необходимость использования передачи сообщений (message passing)
- Сложность реализации некоторых структур данных и алгоритмов
- Высокая латентность и низкая скорость обмена данными между узлами
- Неоднородность, отказы узлов

Классификация М. Флинна,

SISD	Single Instruction – Single Data
MISD	Multiple Instruction – Single Data
SIMD	Single Instruction – Multiple Data
MIMD	Multiple Instruction – Multiple Data

Классификация М. Флинна,



Основные

- Векторно-конвейерные
- Массово-параллельные
- Симметричные мультипроцессоры (SMP)
- Кластеры

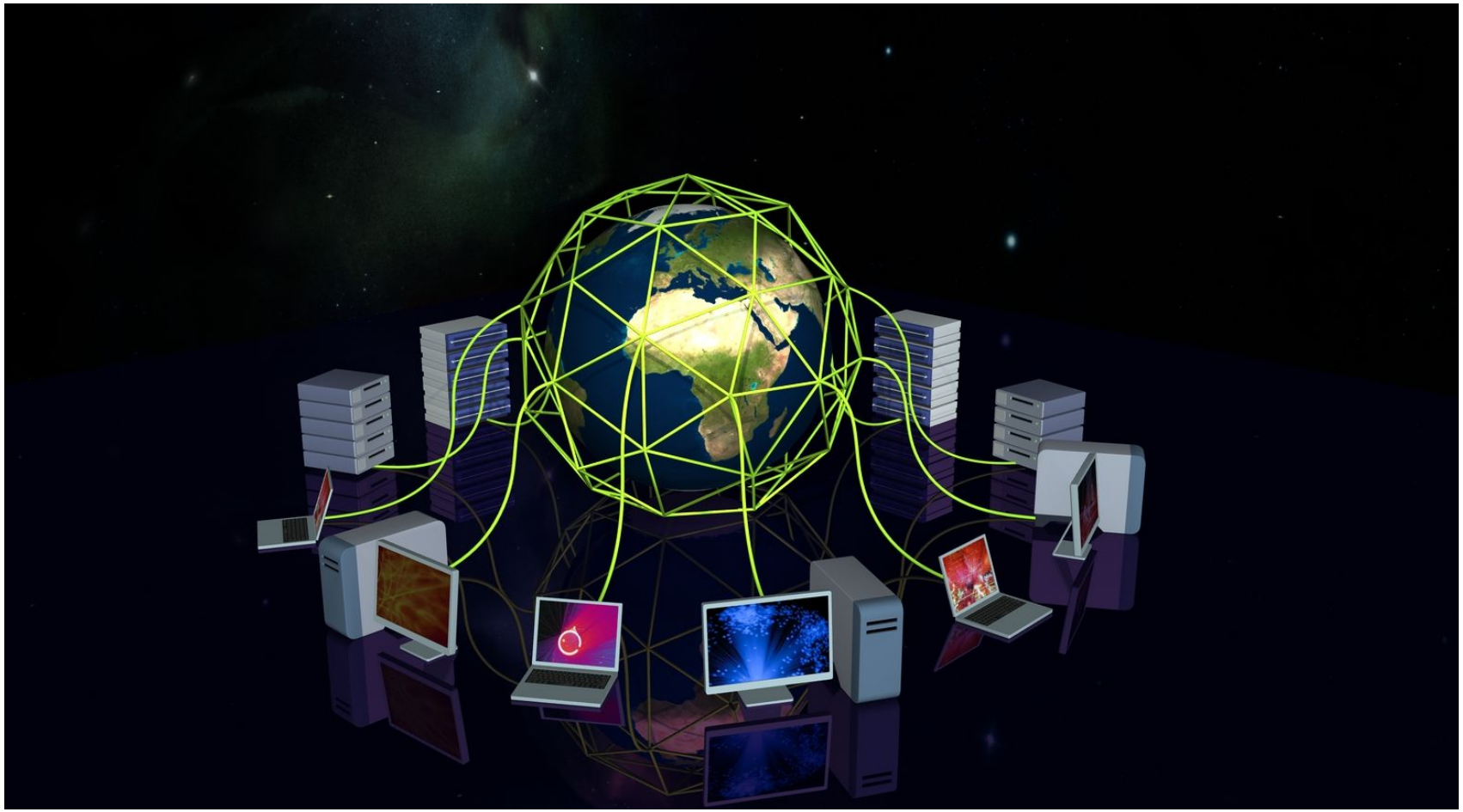
Параллельные векторные системы

Массивно-параллельные системы (MPP)

Симметричные мультипроцессорные системы (SMP)

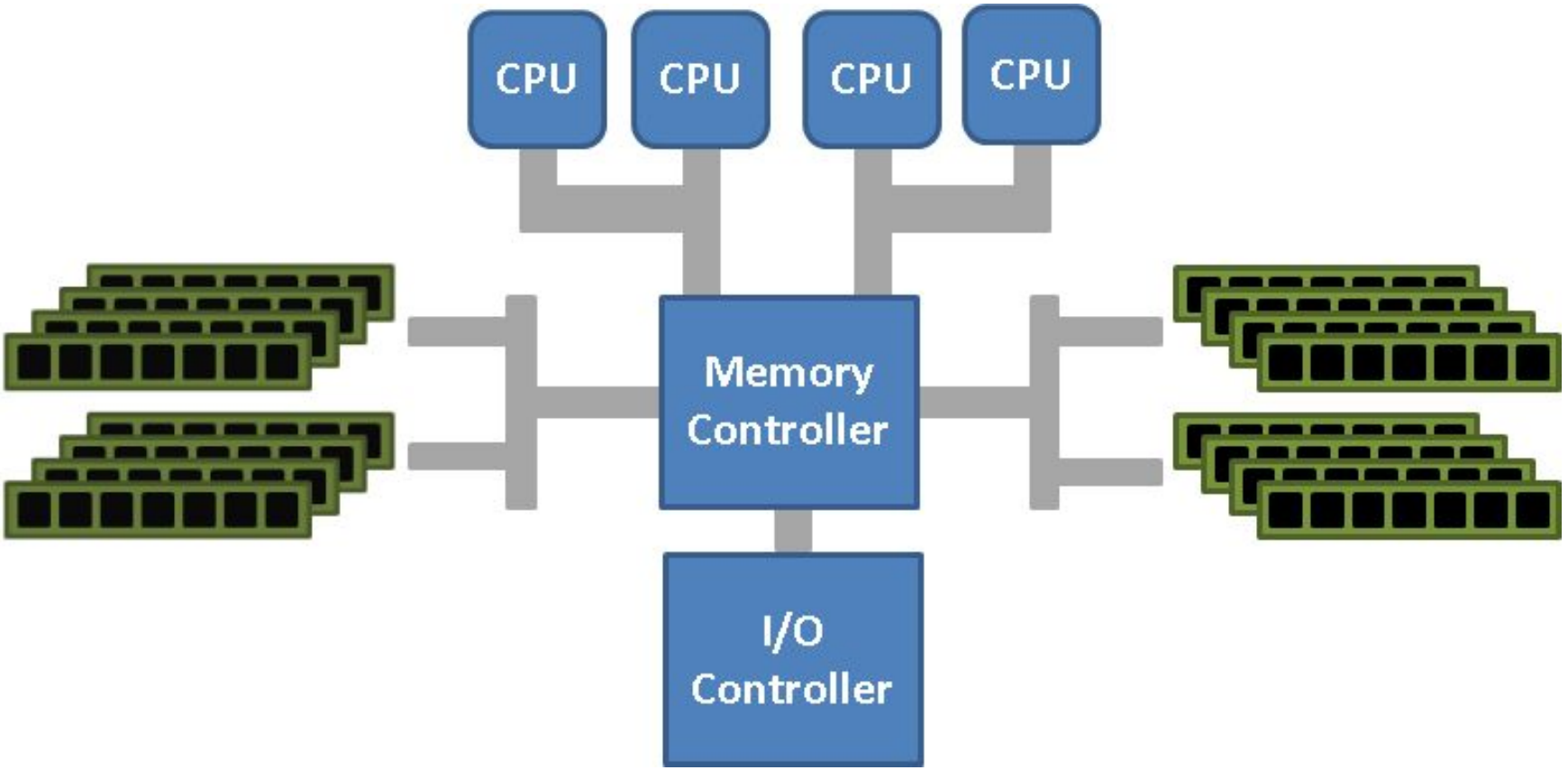
Кластерные системы

Grid (вычислительная сеть)



Графические процессоры (GPU)

Системы с неоднородным доступом к памяти (NUMA)

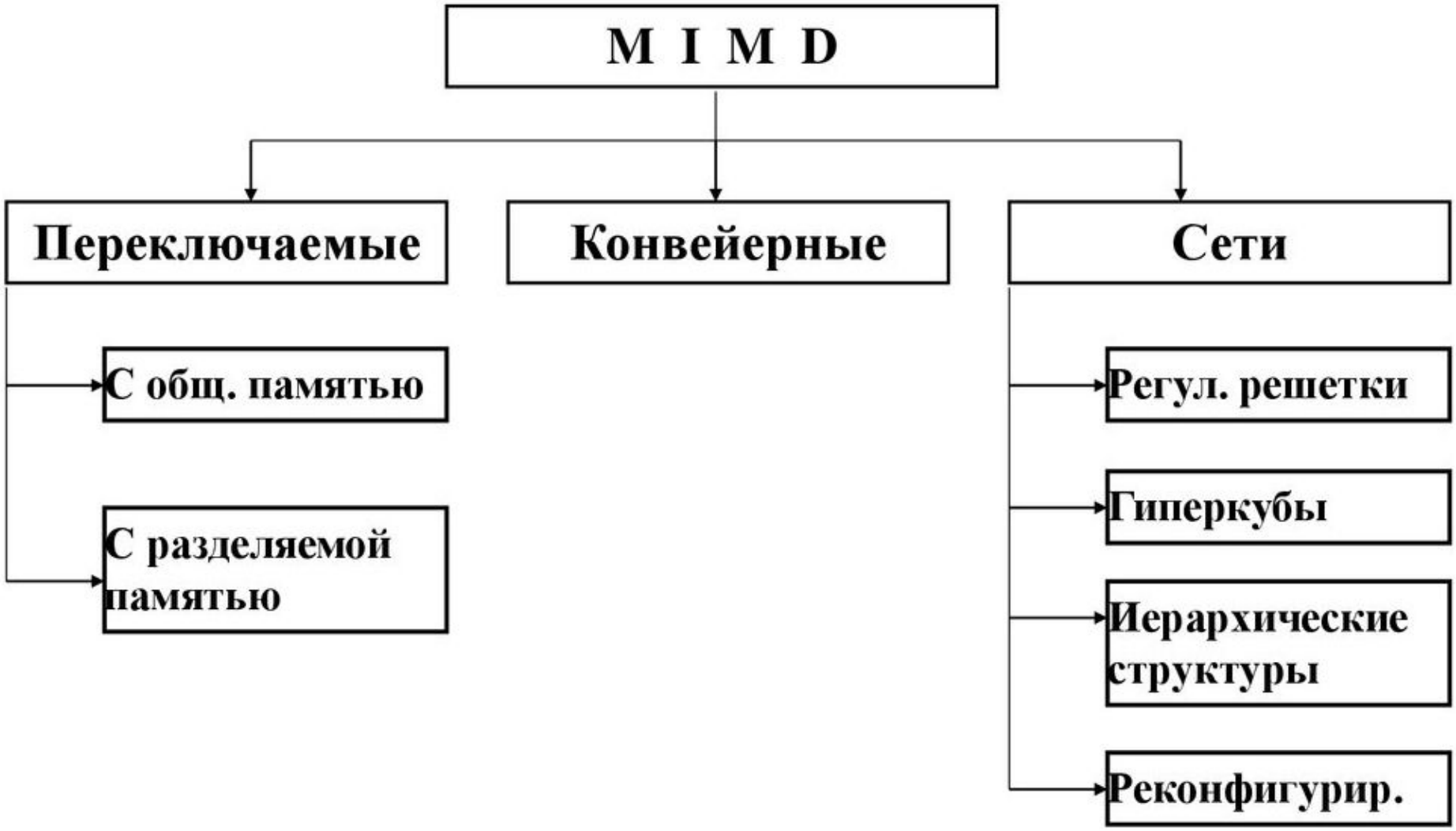


Набор персональных ком

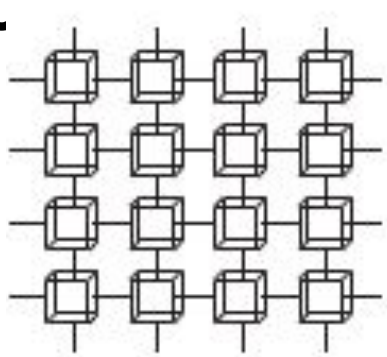
Основные классы современных параллельных ЭВМ

- Компьютеры с распределенной памятью с двухуровневой архитектурой;
- Гибридные метакластерные архитектуры;
- Суперкомпьютеры, использующие многосокетные узлы с многоядерными микропроцессорами в сокетах

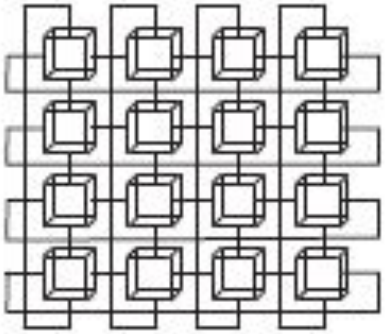
Систематизация MIMD-компьютеров по Р. Хокни



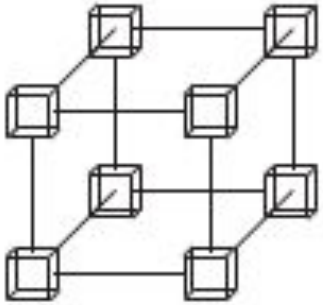
Топологии соединения вычислительных узлов в высокопроизводительных вычислительных с



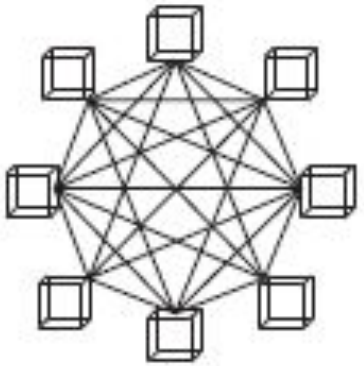
Сетка



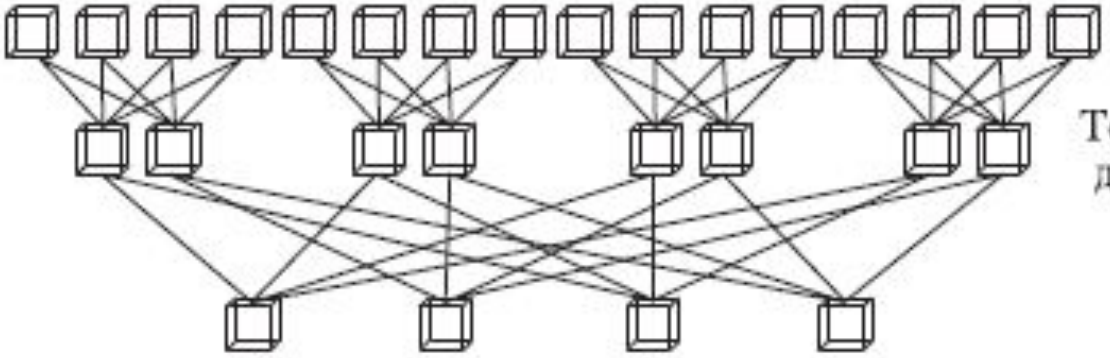
Двухмерный тор



Гиперкуб



Каждый с каждым



Толстое дерево

**Параллельное программирование с использованием OpenMP: учебное пособие / М.П. Левин. – М: Интернет-Университет ИТ, 2008*

классификация Т.

Основана на двух характеристиках:

- число **n** бит в машинном слове, обрабатываемых параллельно;
- число слов **m**, обрабатываемых одновременно вычислительной системой.

P=m×n - максимальная степень параллелизма вычислительной системы

- разрядно-последовательные, пословно-последовательные (n=1, m=1);
- разрядно-параллельные, пословно-последовательные (n>1, m=1);
- разрядно-последовательные, пословно-параллельные (n=1, m>1);
- разрядно-параллельные, пословно-параллельные (n>1, m>1).

Три уровня обработки данных:

- уровень выполнения программы;
- уровень выполнения команд;
- уровень битовой обработки

классификация В.

k - число процессоров;
k’ - глубина макроконвейера;
d - число АЛУ в каждом процессоре;
d’ - глубина конвейера из функциональных устройств АЛУ;
w - число разрядов в слове, обрабатываемых в АЛУ параллельно;
w’ - число ступеней в конвейере функциональных устройств каждого АЛУ.

классификации Д.

- **процессор команд** (IP – Instruction Procesor) – интерпретатор команд;
- **процессор данных** (DP – Data Procesor) – устройство обработки данных;
- **устройство памяти** (IM – Instruction Memory, DM – Data Memory);
- **переключатель** – абстрактное устройство, обеспечивающее связь между процессорами и памятью.

классификации Д.

четыре типа переключателей:

- **1–1** – связывает пару функциональных устройств;
- **n–n** – реализует попарную связь каждого устройства из одного множества с соответствующим ему устройством из другого множества;
- **1–n** – соединяет одно выделенное устройство со всеми функциональными устройствами из некоторого набора;
- **n×n** – каждое функциональное устройство одного множества может быть связано с любым устройством из некоторого набора

Анализ производительности и эффективности параллельных вычислений

Способы параллельной обработки данных:

параллелизм

конвейерность

Свойства параллельных вычислений

Ускорение

$$S_p(n) = \frac{T_1(n)}{T_p(n)}$$

T_1 - время выполнения программы одним процессором

T_p - время выполнения программы конечным числом процессоров

Эффективность

$$E_p(n) = \frac{S_p(n)}{p}$$

Стоимость

$$C_p = pT_p$$

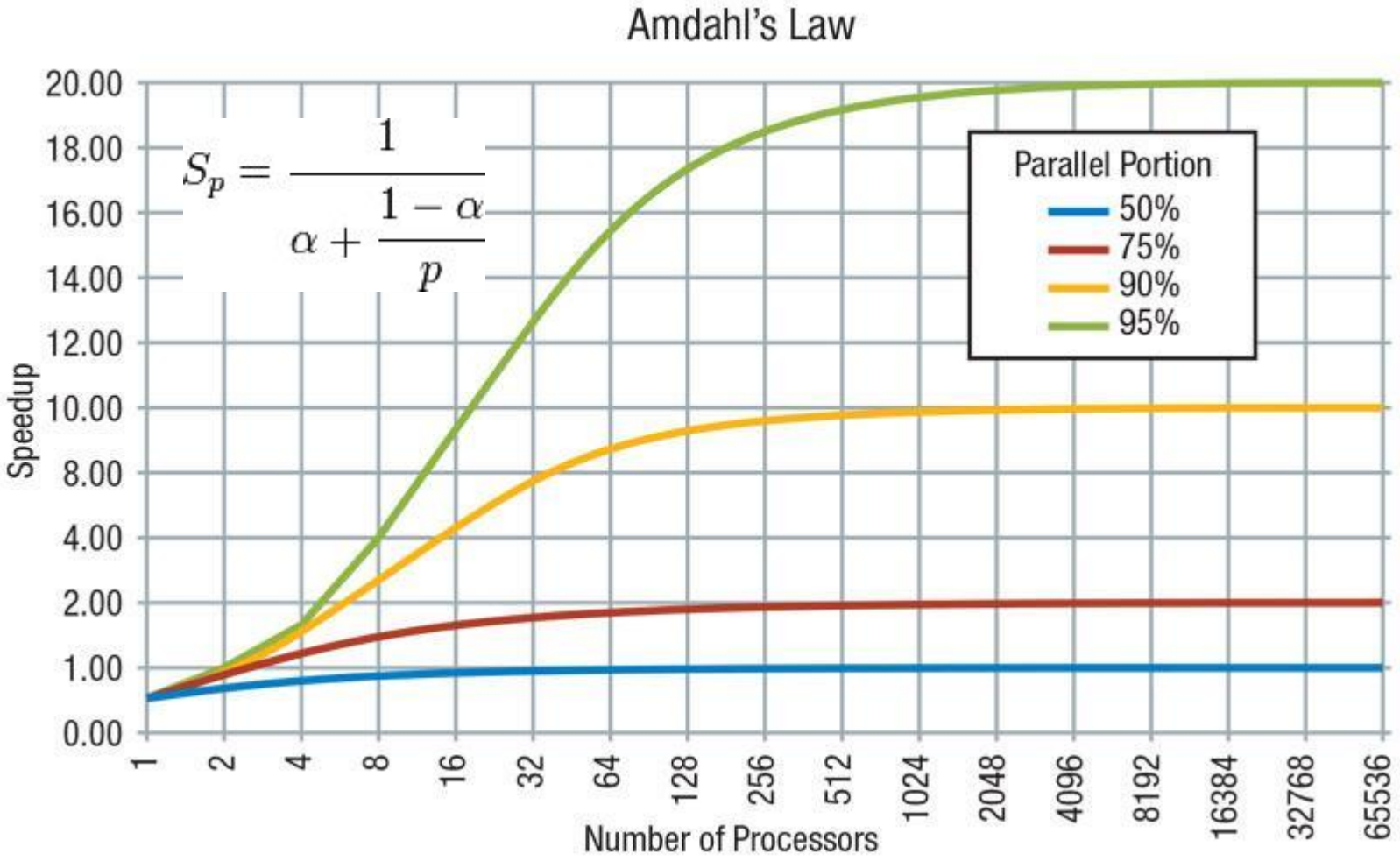
Закон

$$S_p \le \frac{1}{q + \frac{1-q}{p}}$$

q – доля последовательных вычислений в применяемом алгоритме обработки данных,

p – число процессоров

ПОКАЗАТЕЛИ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ



Закон Густафсона-

$$S_n = n + (1 - n)\alpha$$

где α - доля последовательных расчётов в программе, n- количество процессоров.

Масштабируемос

Параллельный алгоритм называют ***масштабируемым*** (*scalable*), если при росте числа процессоров он обеспечивает увеличение ускорения при сохранении постоянного уровня эффективности использования процессоров

$T_0 = pT_p - T_1$

$$T_p = \frac{T_1 + T_0}{p}, \quad S_p = \frac{T_1}{T_p} = \frac{pT_1}{T_1 + T_0}.$$

$$E_p = \frac{S_p}{p} = \frac{T_1}{T_1 + T_0} = \frac{1}{1 + T_0/T_1}.$$

Принципы разработки параллельных алгоритмов (parallel computing)

Классификация алгоритмов по типу

1. **Алгоритмы, использующие параллелизм данных (Data Parallelism).**

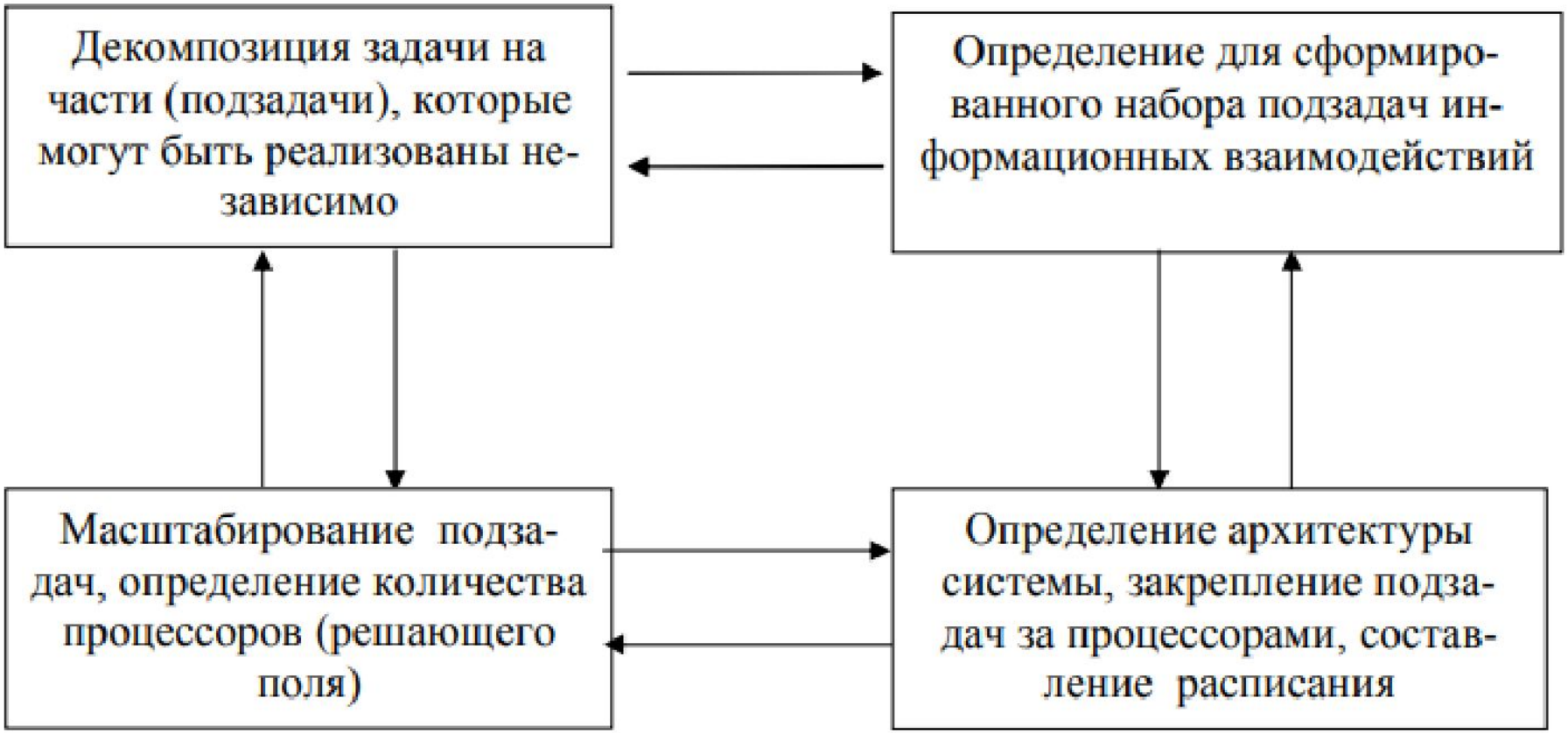
2. **Алгоритмы с распределением данных (Data Partitioning).**

3. **Релаксационные алгоритмы (Relaxed Algorithm).**

4. **Алгоритмы с синхронизацией итераций (Synchronous Iteration).**

Ключевые шаги разработки параллельного алгоритма:

- Поиск параллелизма в известном последовательном алгоритме, его модификация или создание нового алгоритма
- Декомпозиция задачи на подзадачи, которые могут выполняться параллельно
- Анализ зависимостей между подзадачами



Общая схема взаимосвязи этапов разработки параллельных алгоритмов

Специфические задачи реализации параллельного алгоритма в виде параллельной программы:

- Распределение подзадач между процессорами (task mapping, load balancing)
- Организация взаимодействия подзадач (message passing, shared data structures)

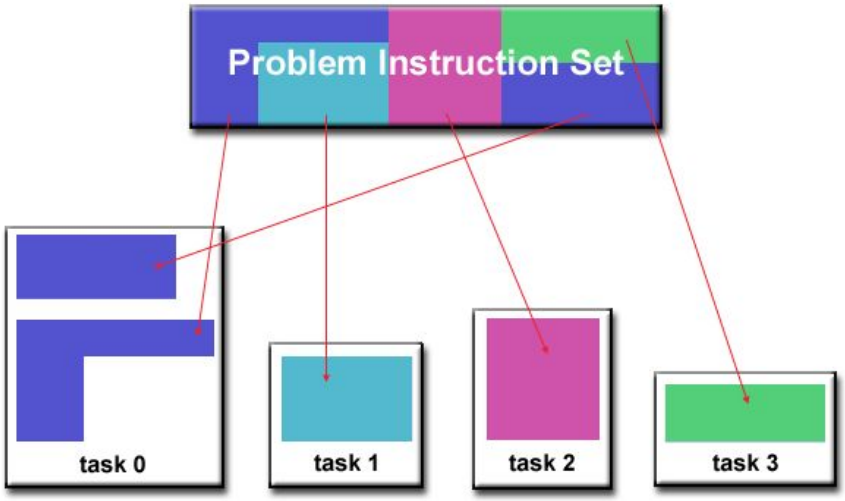
Создание эффективной параллельной реализации алгоритма требует:

- Учета архитектуры целевой вычислительной системы
- Измерения и анализа показателей эффективности

ПОДХОДЫ К ДЕКОМПОЗИЦИИ НА ПОДЗАДАЧИ

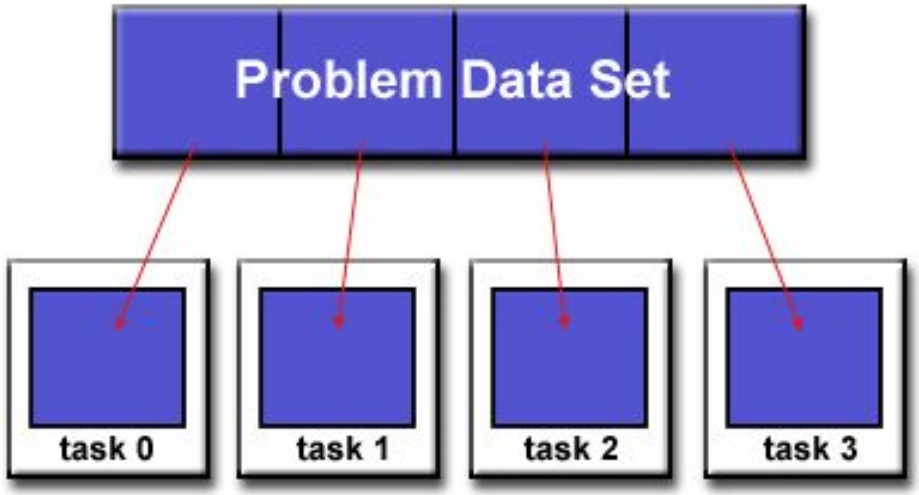
Есть два основных подхода к декомпозиции задач на параллелизуемые подзадачи:

Функциональная декомпозиция
(Task/Functional decomposition)



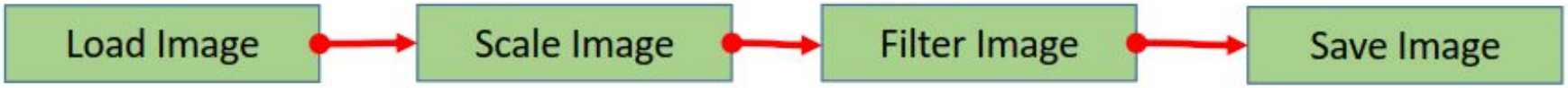
- Распределение вычислений по подзадачам

Декомпозиция по данным
(Domain/Data decomposition)

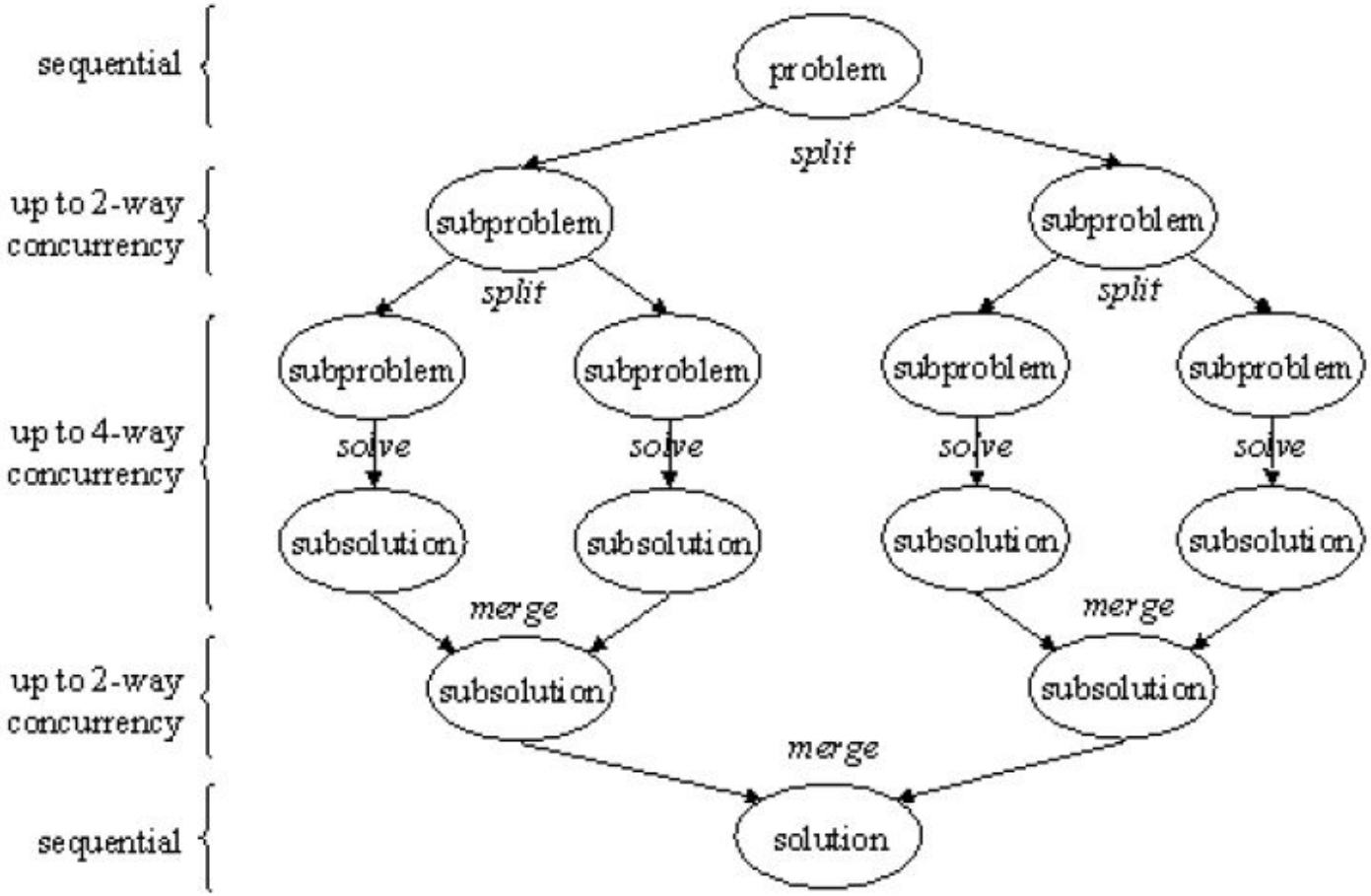


- Распределение данных по подзадачам
- Высокая масштабируемость (многие тысячи ядер)
- Возможность использовать недорогие массовые компоненты (CPU, RAM, сети)

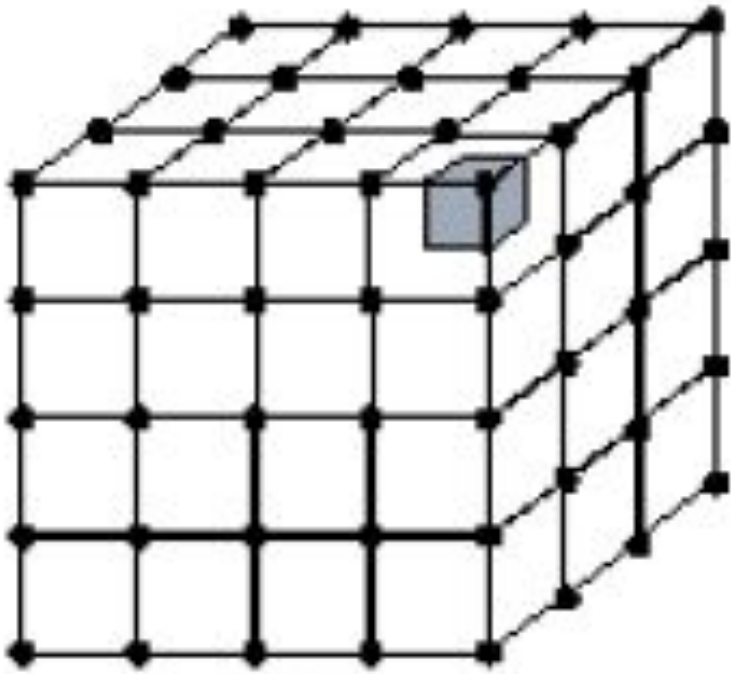
Конвейерная обработка данных



Рекурсивный параллелизм (разделяй и властвуй)



Геометрическая декомпозиция



Разделяемая память (shared memory):

- Аналогия - доска объявлений
- Подзадачи используют общее адресное пространство (оперативной памяти)
- Подзадачи взаимодействуют асинхронно читая и записывая информацию в общем пространстве
- Реализация: многопоточные приложения, OpenMP

Передача сообщений (message passing):

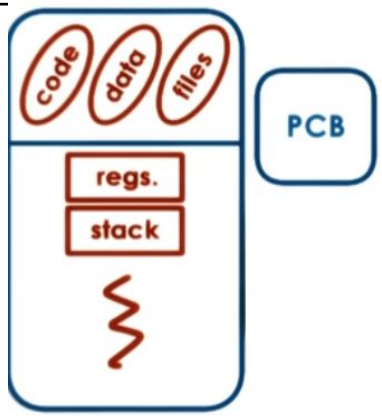
- Аналогия – отправка писем с явным указанием отправителя и получателя
- Каждая подзадача работает с собственными локальными данными
- Подзадачи взаимодействуют за счет обмена сообщениями
- Реализация: MPI (message passing interface)

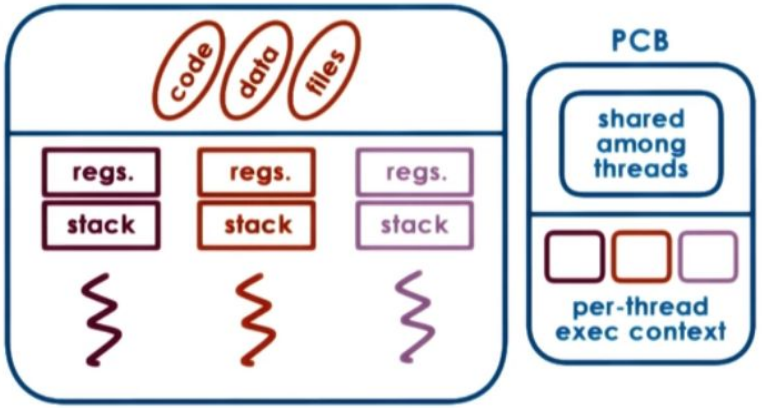
Параллельная обработка данных (data parallelization):

- Строго описанные глобальные операции над данными (может обозначаться как чрезвычайная параллельность (embarrassingly parallel) – очень хорошо распараллеливаемые вычисления)
- Обычно данные равномерно разделяются по подзадачам
- Подзадачи выполняются как последовательность независимых операций
- Реализация может быть сделана как с помощью разделяемой памяти, так и с помощью передачи

ПРОЦЕССЫ И ПОТОКИ

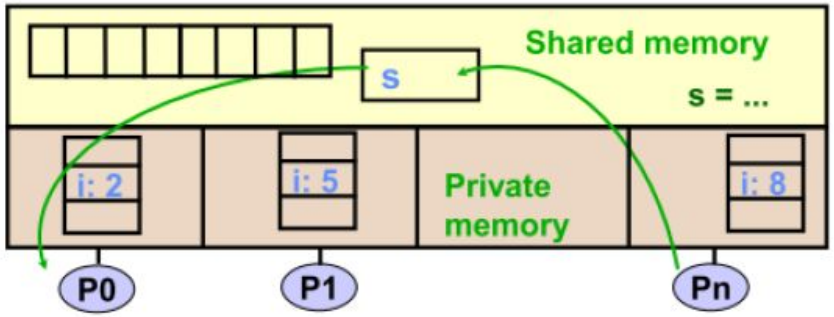
Процесс (process)	**Поток (thread)**
Процесс – полноценная программа (использует большое количество системных ресурсов)	Поток – часть процесса (создается существенно проще)
Разные процессы имеют изолированное адресное пространство	Потоки одного процесса разделяют общую память (и другие ресурсы)
Процессы взаимодействуют через системные механизмы межпроцессной коммуникации	Потоки взаимодействуют через разделяемую память





Сценарий работы многопоточного приложения:

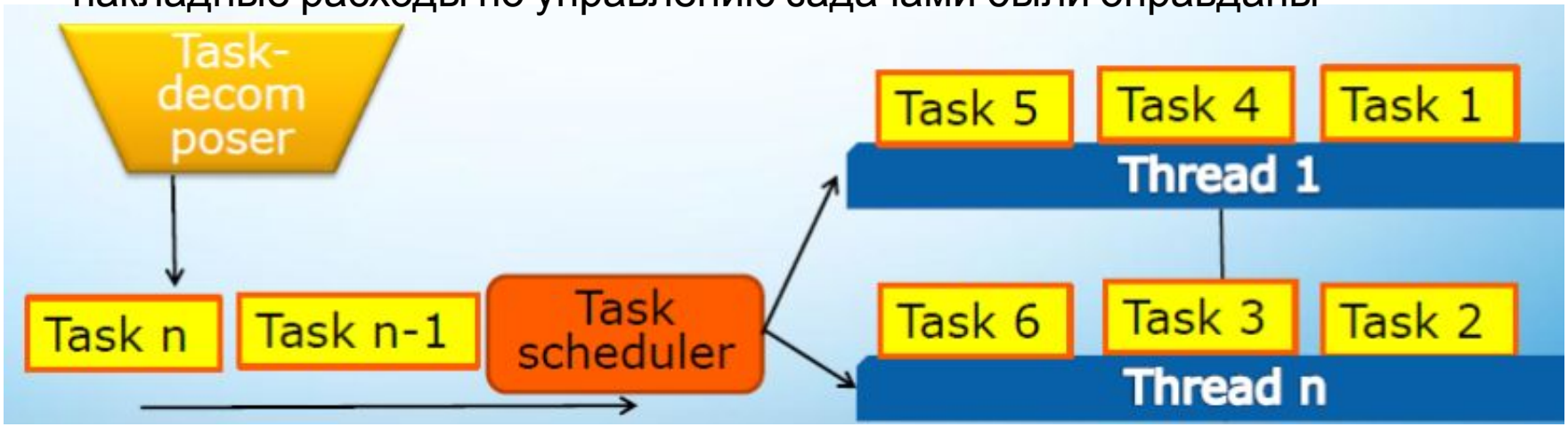
- Сначала инициализируется стартовый поток процесса, выполняет загрузку необходимых ресурсов
- Стартовый поток выполняет последовательные задачи и затем создает новые потоки, выполняемые внутри общего процесса, для параллельного выполнения подзадач
- Каждый поток имеет собственные данные, но разделяет общие ресурсы процесса
- Потоки взаимодействуют через глобальную память процесса (корректное изменение общих данных требует использования механизмов синхронизации)
- Параллельная работа потоков может быть представлена как параллельное исполнение процедур внутри общей программы.



ПОДЗАДАЧИ И ПОТОКИ

- Задача (подзадача) состоит из данных и процедуры их обработки
- Планировщик задач назначает задачу для исполнения в одном из потоков
- Оперирование задачами намного более простое, чем потоками
- Работа планировщика позволяет балансировать нагрузку между потоками

- Задач должно быть намного больше, чем потоков: это обеспечивает гибкость назначения задач и простоту балансировки
- Объем вычислений в задаче должен быть достаточно большим, чтобы накладные расходы по управлению задачами были оправданы



ПРОБЛЕМЫ СИНХРОНИЗАЦИИ

Взаимная блокировка (deadlock) – ситуация в многозадачной среде при которой несколько потоков находятся в состоянии ожидания ресурсов, занятых друг другом, и ни один из них не может продолжать свое выполнение

- Типичная взаимная блокировка: два потока ожидают окончания друг друга

Состояние гонки (race condition) – ситуация в которой работа приложения зависит от того, в каком порядке выполнятся (параллельные) части кода

- Несколько потоков модифицируют разделяемый ресурс (например, переменную)
- Результат зависит от того, какой поток первым выполнит изменения
- Проблема может быть решена за счет блокировок, но зачастую это сложно, и приводит к ошибкам, в частности, взаимной блокировке

Потоковая безопасность (thread safety) – специфика кода (например функций или библиотек), позволяющая использовать его из нескольких потоков одновременно

- Источником нарушения потоковой безопасности может быть: доступ к глобальным переменным или динамической памяти; выделение/освобождение глобальных ресурсов (например файлов); неявный доступ через указатели; побочный эффект функций.

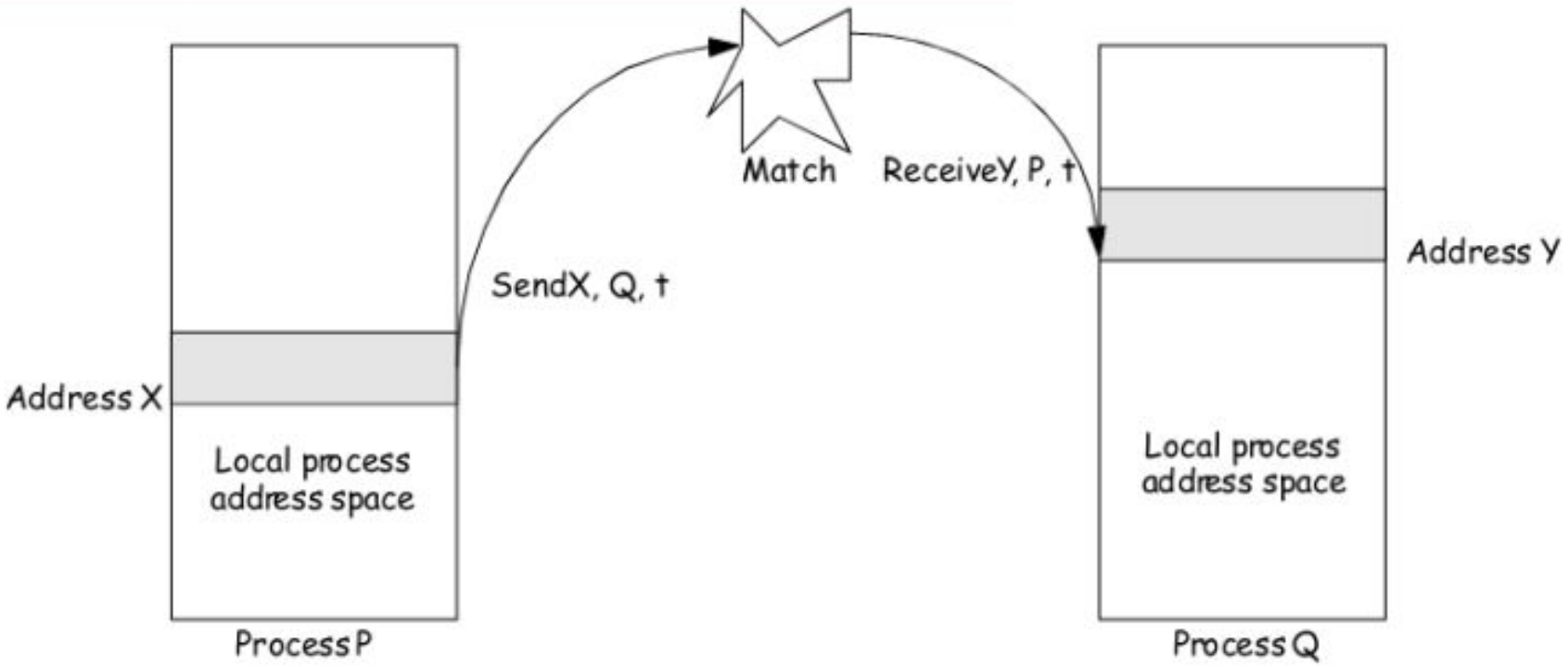
Основные характеристики модели на основе передачи сообщений:

- Набор задач, имеющих свою собственную локальную память во время вычислений
- Задачи могут находится как на одной машине (в т.ч. с разделяемой памятью), так и на разных машинах
- Задачи обмениваются данными с помощью отсылки и приема сообщений явно описываемых в программном коде
- Зачастую передача данных подразумевает их сериализацию/десериализацию, что требует соответствующих накладных расходов
- Как правило передача данных требует совместной работы, выполняемой как задачей-отправителем, так и задачей-получателем

Программирование для модели на основе передачи сообщений:

- С точки зрения программирования модель на основе передачи сообщений выглядит как внедрение вызовов специализированной библиотеки в программный код.
- За реализацию параллелизма полностью отвечает программист, а не компилятор
- Общепринятым стандартом для модели параллельного программирования на основе передачи сообщений является библиотека MPI (Message Passing Interface).

ПРИМЕР ПЕРЕДАЧИ СООБЩЕНИЯ



Параллельная обработка данных (data parallelization):

- Строго описанные глобальные операции над данными (может обозначаться как чрезвычайная параллельность (embarrassingly parallel) – очень хорошо распараллеливаемые вычисления)
- Обычно данные равномерно разделяются по подзадачам
- Подзадачи выполняются как последовательность независимых операций
- Реализация может быть сделана как с помощью разделяемой памяти, так и с помощью передачи сообщений

Основные характеристики модели на основе параллельной обработки данных:

- Основные параллельные задачи сфокусированы на выполнении операций над неким массивом данных
- Массив данных обычно организован в виде однородной структуры, например массива или гиперкуба
- Задачи обычно параллельно выполняют аналогичные операции над выделенными им фрагментами одного массива данных
- В реализации на архитектурах без разделяемой памяти массив данных делится на фрагменты, которые находятся в распоряжении отдельных задач
- Программирование для данной модели обычно представляет собой написания программы оперирующей с конструкциями для параллельной обработки данных, например в виде вызовов специализированной библиотеки

Ресурсы для самоподготовки:

1. Малявко, А. А. Параллельное программирование на основе технологий openmp, mpi, cuda : учебное пособие для академического бакалавриата / А. А. Малявко. — 2-е изд., испр. и доп. — Москва : Юрайт, 2019. — 129 с. - ЭБС Юрайт. — URL: https://biblio-online.ru/bcode/446247 (дата обращения: 17.01.2020). —Текст : электронный.

2. Карепова, Е.Д. Основы многопоточного и параллельного программирования: учебное пособие / Е.Д. Карепова. - Красноярск: Сиб. Федер. ун-т, 2016. — 356 с. — ЭБС ZNANIUM.com. — URL: https://new.znanium.com/catalog/product/966962 (дата обращения: 17.01.2020). — Текст : электронный.

3. Федотов, И.Е. Модели параллельного программирования: практическое пособие / И.Е. Федотов. — Москва: СОЛОН-Пресс, 2017. — 392 с. — ЭБС ZNANIUM.com. — URL: https://new.znanium.com/catalog/product/858609 (дата обращения: 17.01.2020). — Текст : электронный.

СПАСИБО ЗА ВНИМАНИЕ!

ФИНАНСОВЫЙ УНИВЕРСИТЕТ
ПРИ ПРАВИТЕЛЬСТВЕ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Корчагин Сергей Алексеевич
SAKorchagin@fa.ru
2021