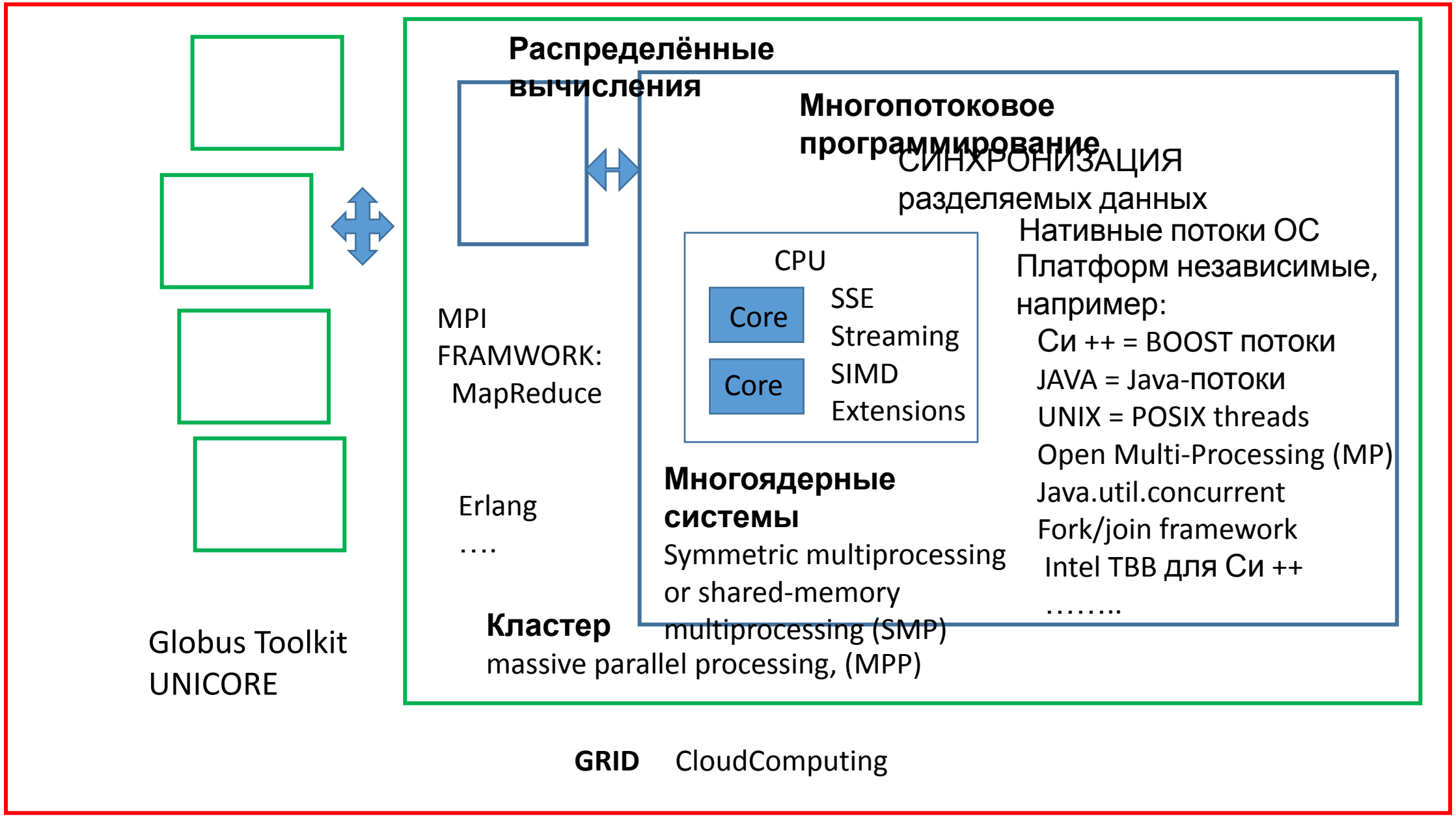

Распределённые вычисления
Многопотоковое программирование
СИНХРОНИЗАЦИЯ разделяемых данных
Нативные потоки ОС
Платформ независимые, например:
Си ++ = BOOST потоки
JAVA = Java-потоки
UNIX = POSIX threads
Open Multi-Processing (MP)
Java.util.concurrent
Fork/join framework
Intel TBB для Си ++
……..
CPU
Core
Core
SSE
Streaming
SIMD
Extensions
MPI
FRAMWORK:
MapReduce
Erlang
….
Многоядерные системы
Symmetric multiprocessing or shared-memory multiprocessing (SMP)
Кластер
massive parallel processing, (MPP)
Globus Toolkit
UNICORE
GRID
CloudComputing

```
for (i = 0; i < 1024; i++) C[i] = A[i] * B[i];
```

```
for (i = 0; i < 1024; i+=4) C[i:i+3] = A[i:i+3] * B[i:i+3];
```

`C[i:i+3]` означает вектор из 4 элементов — от `C[i]` до `C[i+3]` включительно,
`*` означает операцию поэлементного умножения векторов.

Тогда векторный процессор сможет выполнить 4 скалярные операции при помощи одной векторной инструкции за время, близкое к выполнению скалярной операции.
Таким образом, векторных операций потребуется в 4 раза меньше, и программа исполнится быстрее.

Векторные операции могут добавляться в скалярные процессоры, тогда они называются векторными расширениями команд.
Примеры: MMX, SSE, SSE2, AltiVec.

1997 – MMX (Pentium MMX)
8 байт, целочисленные операции

1998 – 3DNow! (K6-2)
8 байт, float

1999 – SSE (Pentium III)
16 байт, float

2001 – SSE2 (Pentium 4)
16 байт, double

2004 – SSE3 (Pentium 4 Prescott)

2006 – SSSE3 (Core 2, Atom)

2007 – SSE4.1 (Penryn)

2008 – SSE4.2 (Nehalem)
Специальные команды: обработка видео, обработка строк, криптография

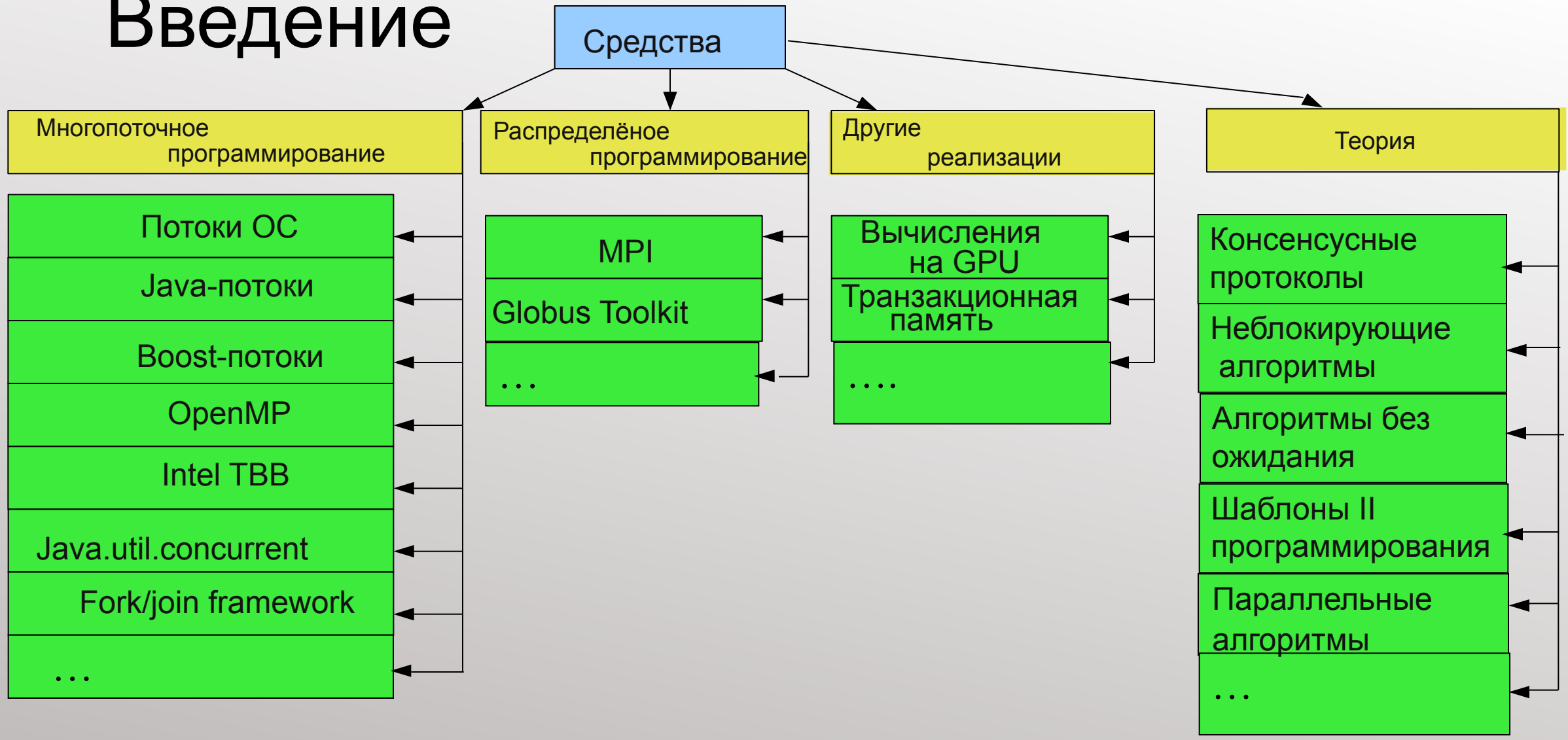
2011 – AVX (Sandy Bridge)
32 байта, float, double
Трёхоперандные команды

2013 – AVX2 (Haswell)
32 байта, целочисленные операции
FMA: r = a*b + c

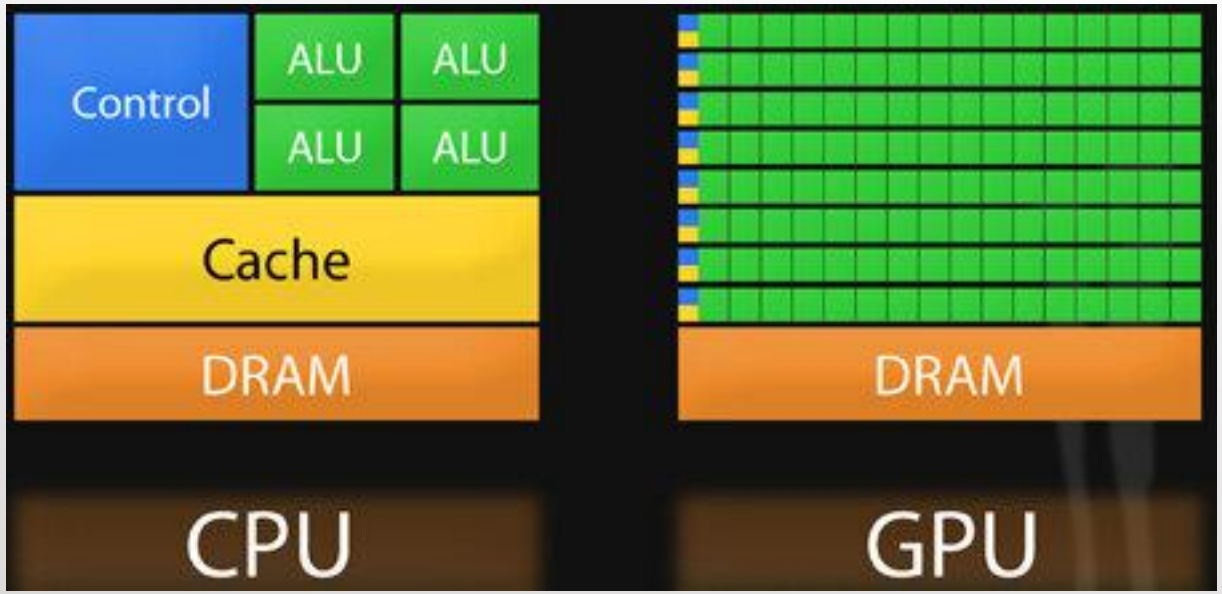
2016 – AVX-512 (KNL)
64 байта

••••••••••••••••••••

Введение


Средства
Многопоточное программирование
Распределённое программирование
Другие реализации
Теория
Потоки ОС
Java-потоки
Boost-потоки
OpenMP
Intel TBB
Java.util.concurrent
Fork/join framework
…
MPI
Globus Toolkit
…
Вычисления на GPU
Транзакционная память
….
Консенсусные протоколы
Неблокирующие алгоритмы
Алгоритмы без ожидания
Шаблоны II программирования
Параллельные алгоритмы
…

GPU — Graphical Processing Unit

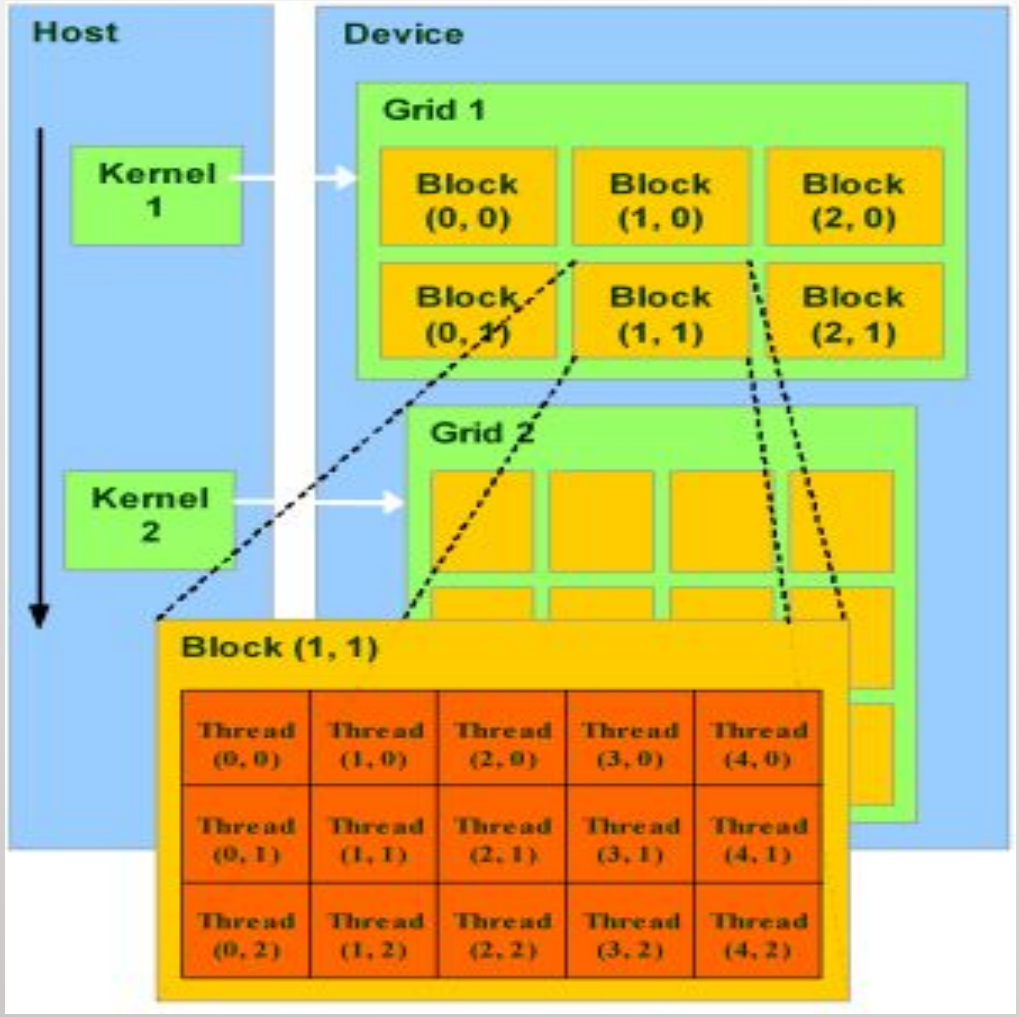


- Меньше транзисторов на управление и кэш
- Больше на АЛУ
- Однотипные операции над большим количеством данных

- CUDA (Compute Unified Device Architecture)
- OpenCL (Open Computing Language) *OpenGL(графика) OpenAL (звук)*

Программная модель CUDA

- Модель SIMD вычислений.
- Потоки объединяются в блоки потоков (thread block) — одно-, двух- или трехмерным сетки потоков, взаимодействующих между собой при помощи разделяемой памяти и точек синхронизации.
- Программа (ядро, kernel) исполняется над сеткой (grid) блоков потоков (thread blocks).
- Одновременно исполняется одна сетка.



слабая переносимость

Транзакционная память (transactional memory) для многопоточного программирования

в ней реализован механизм управления параллельными процессами для обеспечения доступа к совместно используемым данными, обеспечивающий контроль за актуальностью данных в оперативной памяти:

- блокировка данных для записи для одних потоков,
- открытие чтения или изменения данных для других потоков.

Транзакционная память (transactional memory -TM)

Транзакция (Atomicity, Consistency, Isolation, Durability –принципы ASID):

- **Неделимость** — транзакция представляет собой единое целое, не может быть частичной транзакции, если выполнена часть транзакции, она отклоняется.
- **Согласованность** — транзакция не нарушает логику и отношения между элементами данных.
- **Изолированность** — результаты транзакции не зависят от предыдущих или последующих транзакций.
- **Устойчивость** — после своего завершения транзакция сохраняется в системе, происходит фиксация транзакции.

Реализация:

- Программная (Software Transactional Memory, STM)
- Аппаратная (Hardware Transactional Memory, HTM)
- Смешанная

Консенсусные протоколы

- Консенсус – совместное однократное принятие общего решения N потоками из предложенных.
- Есть сеть:
  - с N узлами,
  - все узлы формируют сообщения в произвольный момент времени,
  - сеть доставляет эти сообщения, при этом задержки могут быть случайными,
  - все сообщения приходят в узлы в произвольном порядке с разной задержкой,
  - некоторые узлы могут быть злоумышленниками и мы не знаем кто это.

  Децентрализованный протокол консенсуса:
  - Выполняет упорядочивание этих сообщений и у каждого узла появляется одна и таже копия блокчейна или другой алгебраической структуры.

  Характеристики:

  Пропускная способность, масштабируемость, вычислительная сложность.

Блокчейн (**blockchain)** — это распределенная база данных, которая содержит информацию обо всех транзакциях, проведенных участниками системы. Информация хранится в виде цепочки блоков. В каждом из них записано определенное число транзакций.

Сложная HASH функция

Транзакционная память (transactional memory -TM)

Основа – два сложно реализуемых механизма:

- управление версиями данных (data versioning)
- обнаружение конфликтов (conflict detection)

Принцип «все или ничего»:

- Если транзакцию удается выполнить от начала до конца, то она результативно осуществляется, и в данные заносятся те или иные изменения.
- Если же в процессе выполнения транзакции возникает конфликт, то в таком случае отрабатывается откат назад, возможные изменения в данных восстанавливаются, и все начинается с начала.

Неблокирующие алгоритмы

Формальное определение lock-free объекта звучит так: разделяемый объект называется lock-free объектом (неблокируемым, non-blocking объектом), если он гарантирует, что *некоторый* поток закончит выполнение операции над объектом за конечное число шагов вне зависимости от результата работы других потоков (даже если эти другие потоки завершились крахом).

Проблема – сложность

Неблокирующие алгоритмы

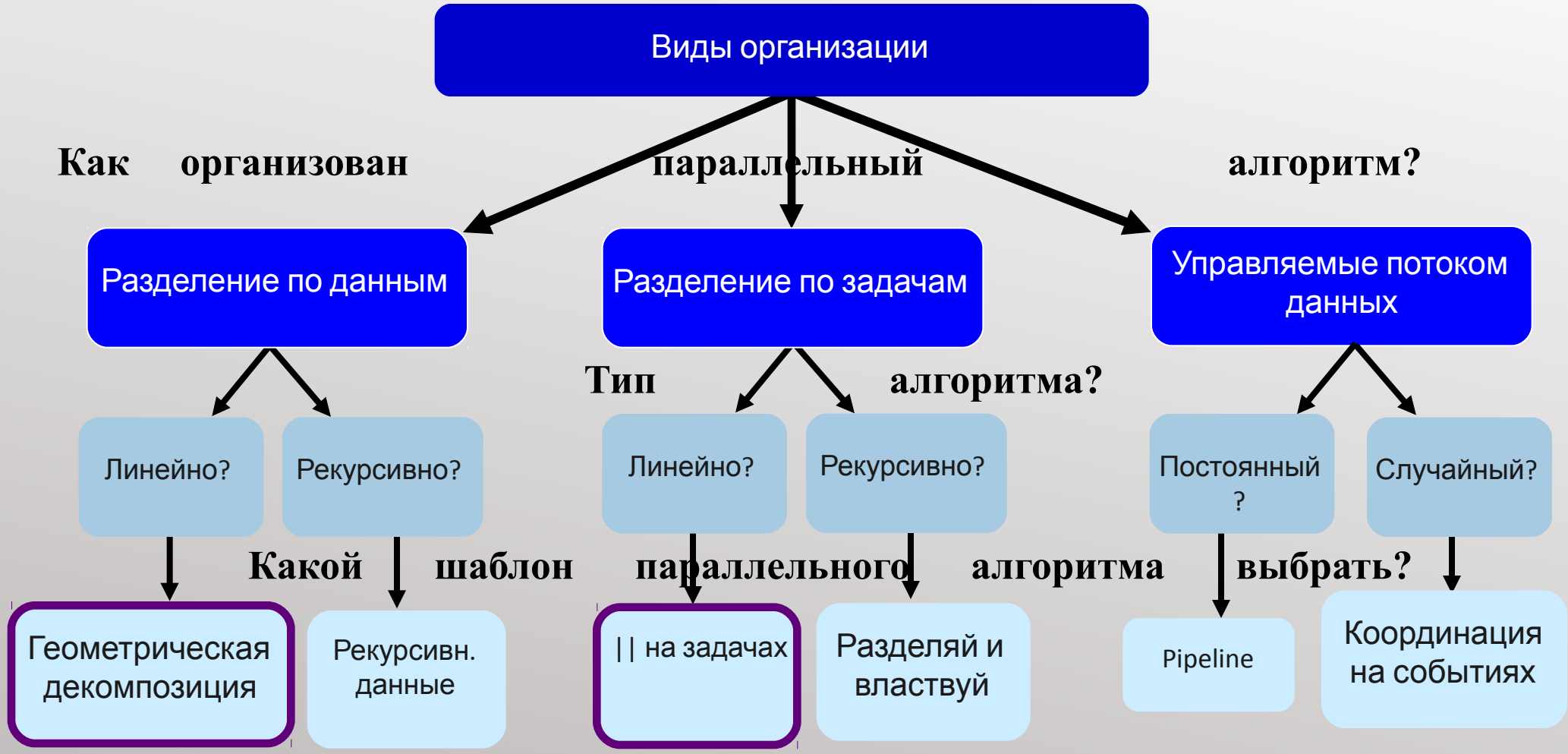
- Неблокирующая синхронизация — подход в параллельном программировании, в котором отходят от традиционных примитивов блокировки, таких, как семафоры, мьютексы и события. Разделение доступа между потоками идёт за счёт атомарных и специальных, разработанных под конкретную задачу, механизмов блокировки.
- Преимущество неблокирующих алгоритмов — в лучшей масштабируемости по количеству процессоров.

Алгоритм называется не блокирующим, если отказ или остановка любого потока не может привести к отказу или остановке любого другого потока.

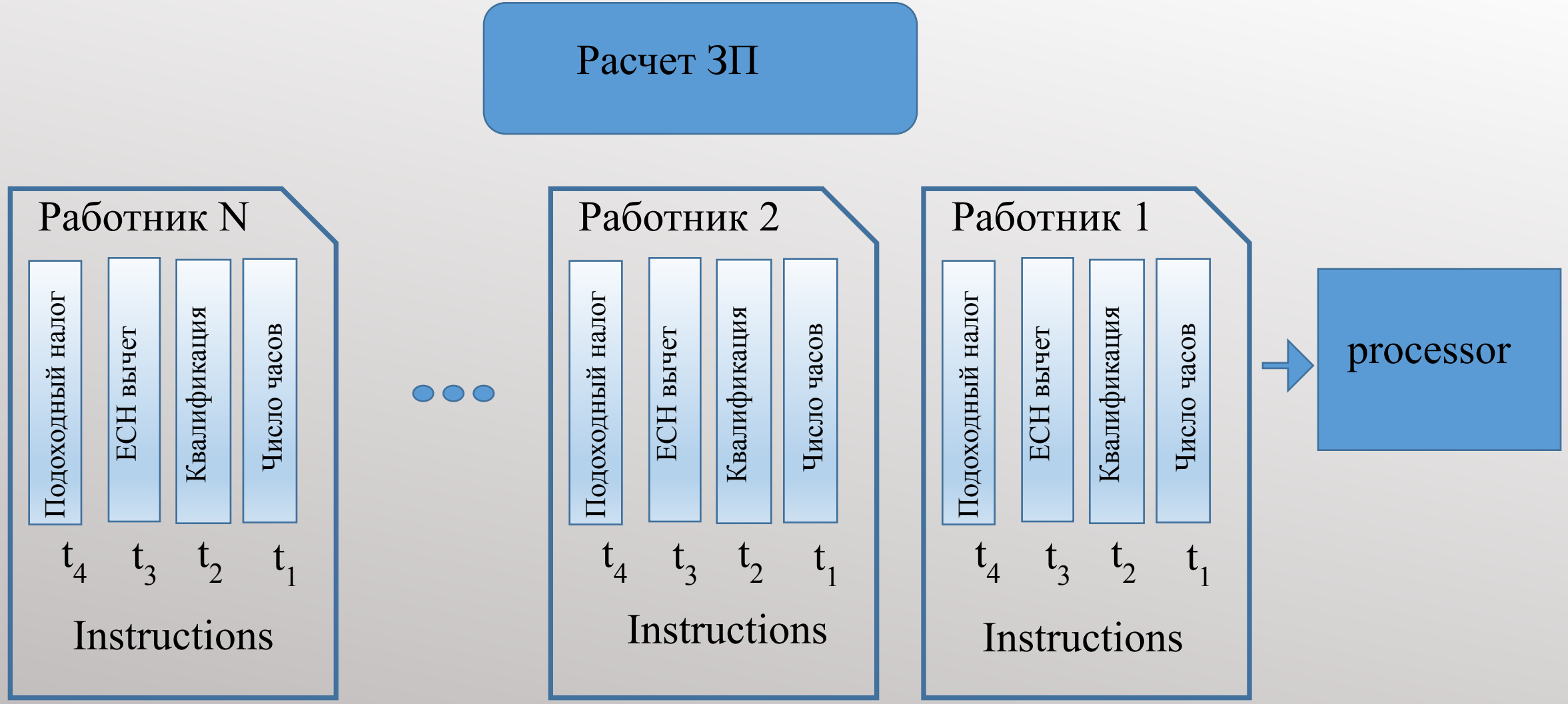
Шаблоны параллельного программирования (*pattern)*

- Существует огромное количество структур организации параллельных программ
- Наиболее часто встречающихся схем :
  - Схема “разделяй и властвуй”
  - Конвейерная схема обработки данных
  - Рекурсивная схема обработки данных
  - Схема, основанная на геометрическом разделении данных
  - Параллелизм задач
  - Параллелизм, основанный на возникновении событий

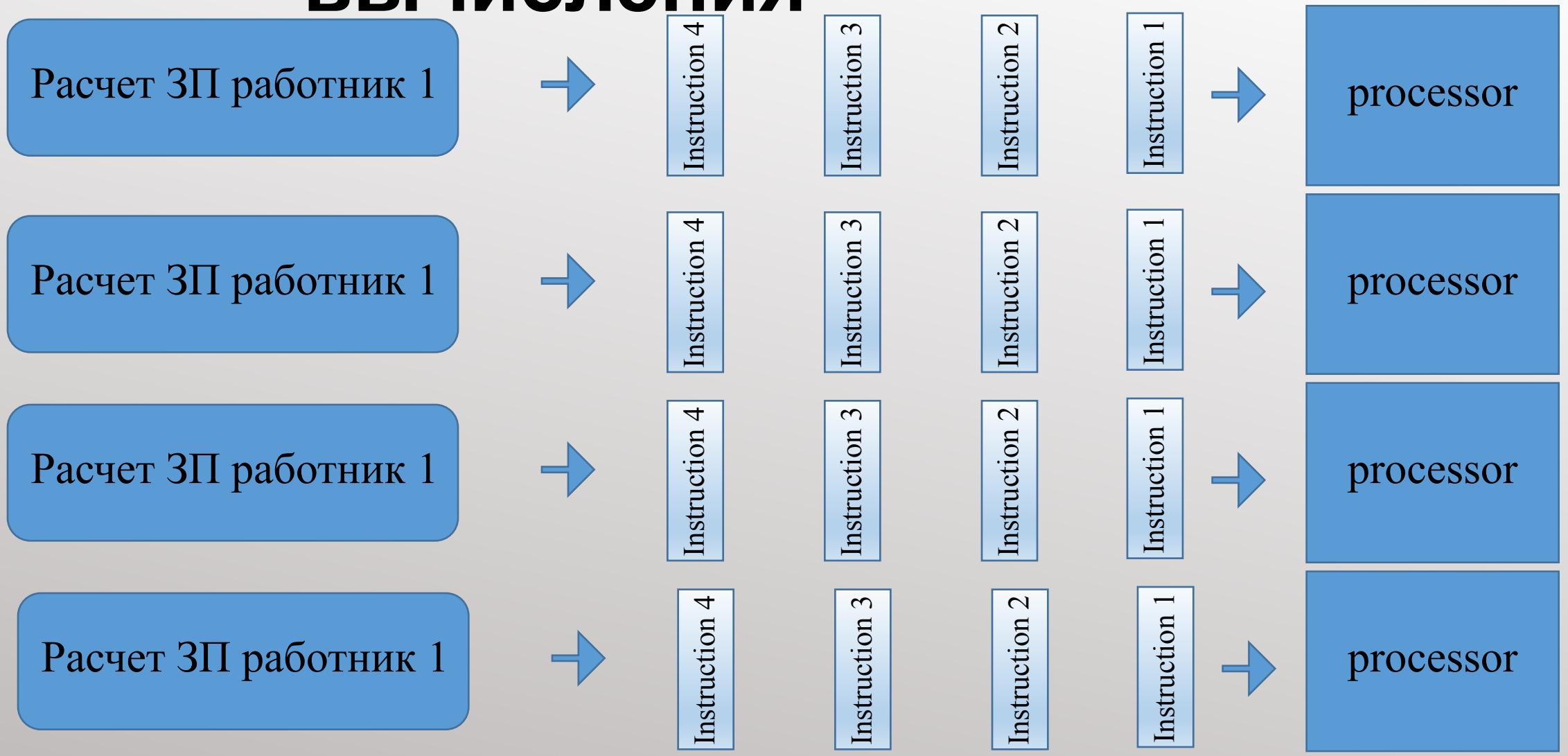
Шаблоны параллельного программирования (*pattern)*



Последовательные вычисления

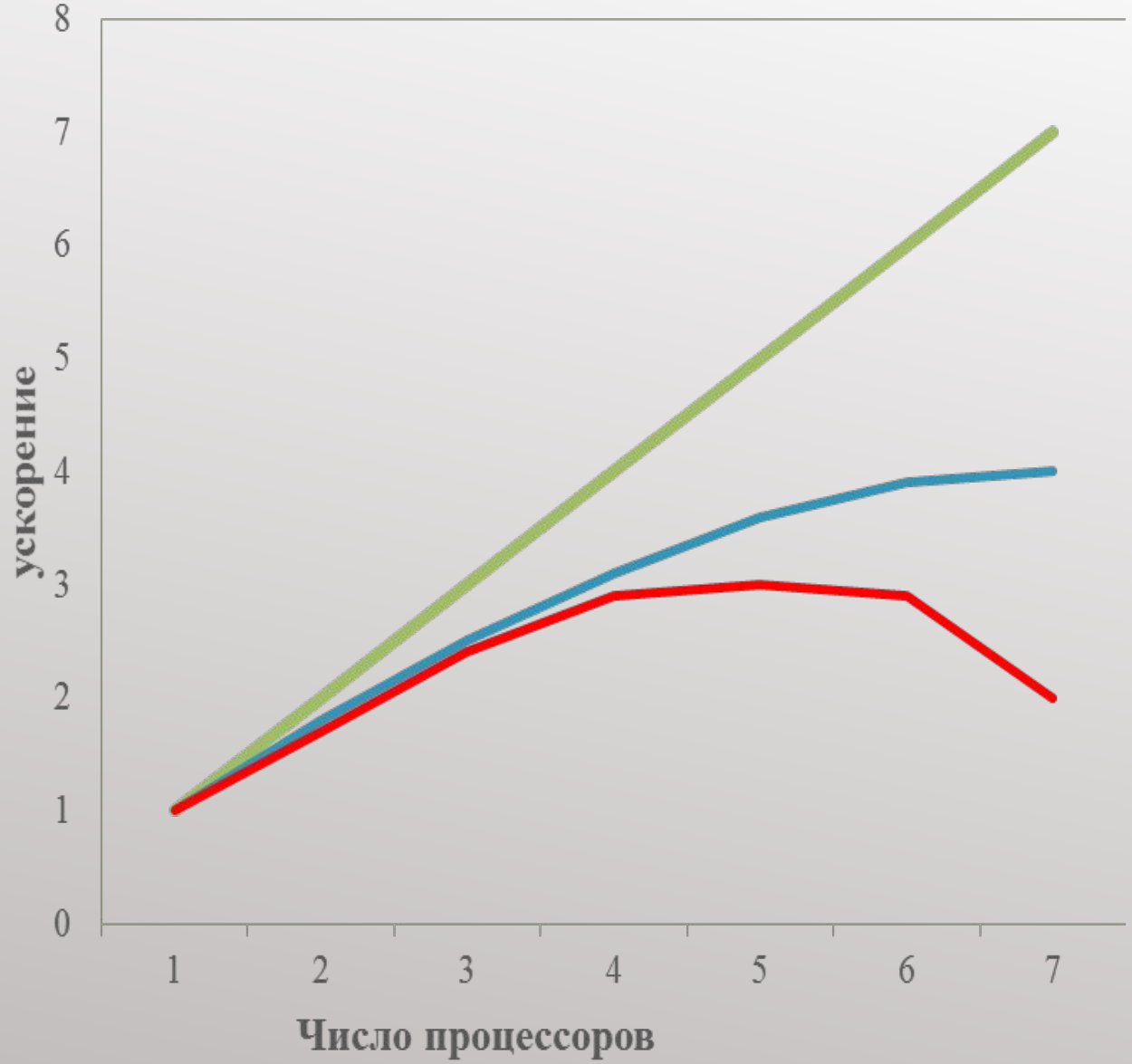


Параллельные вычисления



- Параллельные вычисления — это одновременное использование нескольких вычислений для решения вычислительной задачи:
- Задача разбита на отдельные части, которые можно решать одновременно
- Каждая часть разбита на серию инструкций
- Инструкции из каждой части выполняются одновременно на разных процессорах
- **Используется общий механизм управления/координации.**

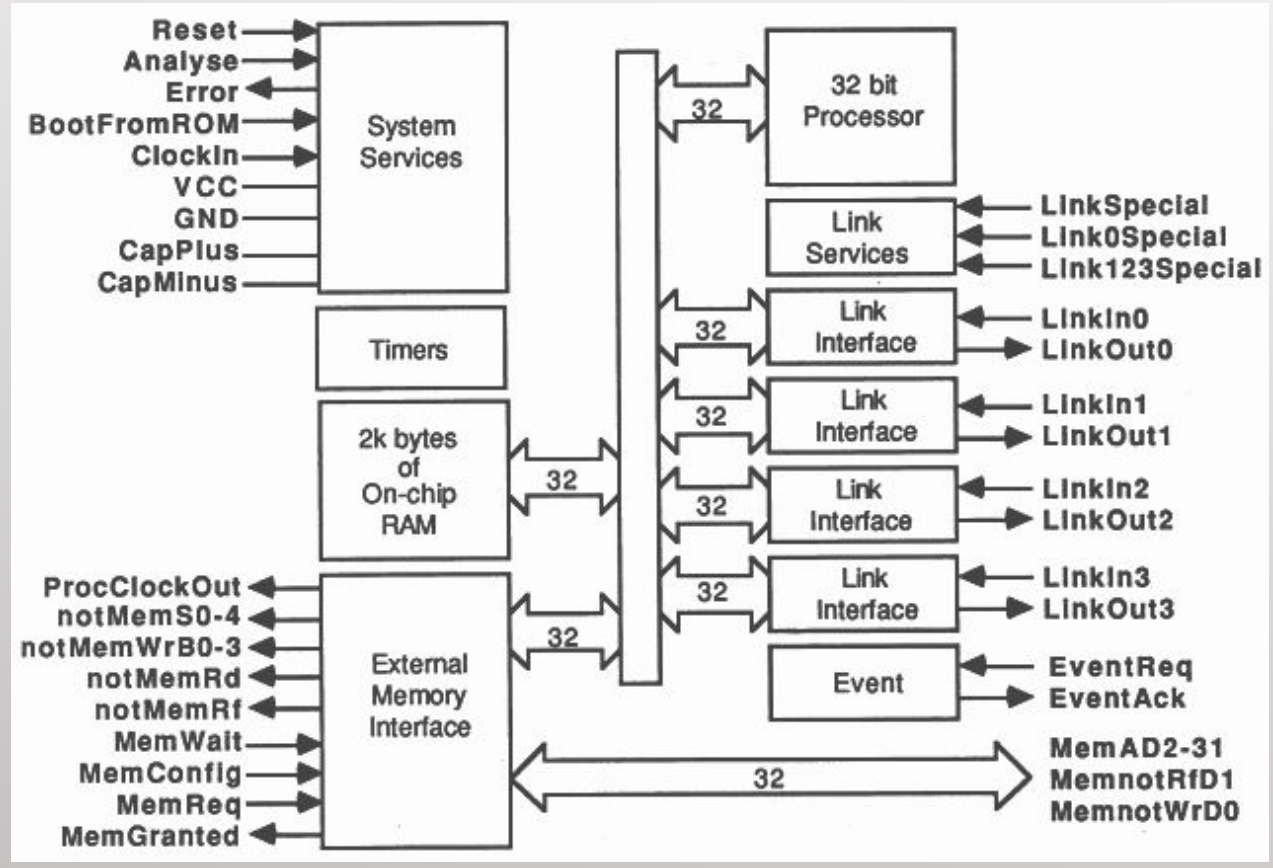
Ускорение (Speedup)



Процессоры нетрадиционной архитектуры

Transputer - инновационный микропроцессор 1980-х годов со встроенной памятью, аппаратным планировщиком и 4-мя последовательными каналами связи.

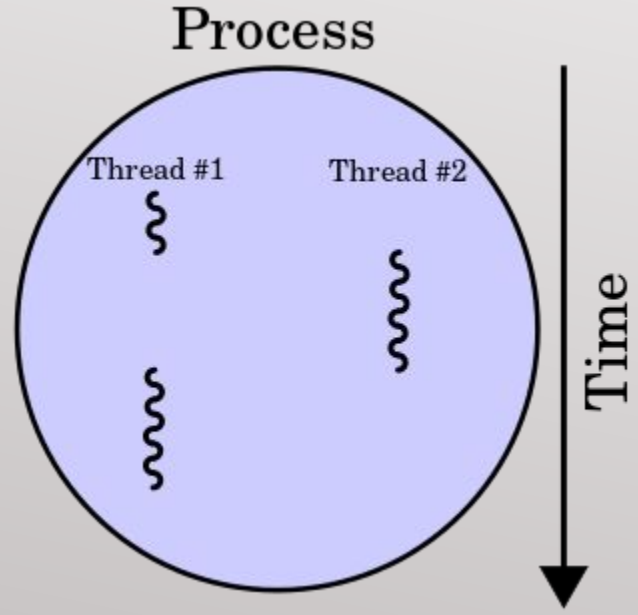
INMOS
Unitted
Kingdom



Thread-level parallelism (TLP)

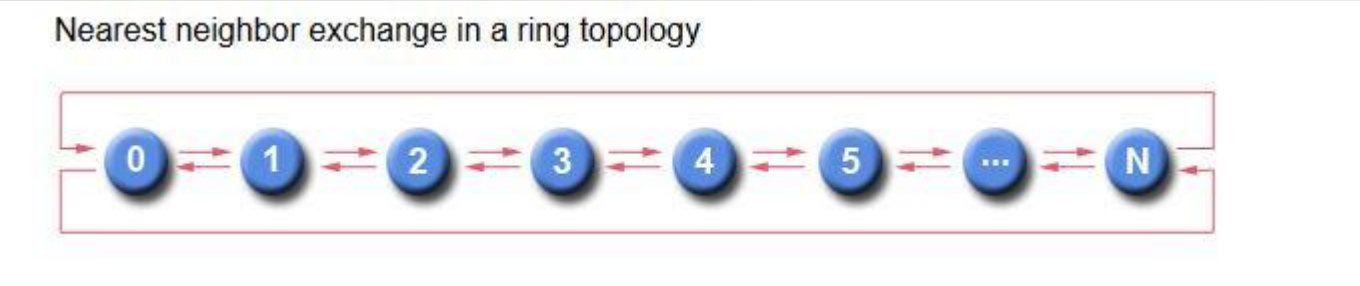
Многопоточность — это способность центрального процессора (ЦП) (или одного ядра в многоядерном процессоре) одновременно выполнять несколько процессов или потоков, поддерживаемых операционной системой или оборудованием (режим разделения времени).

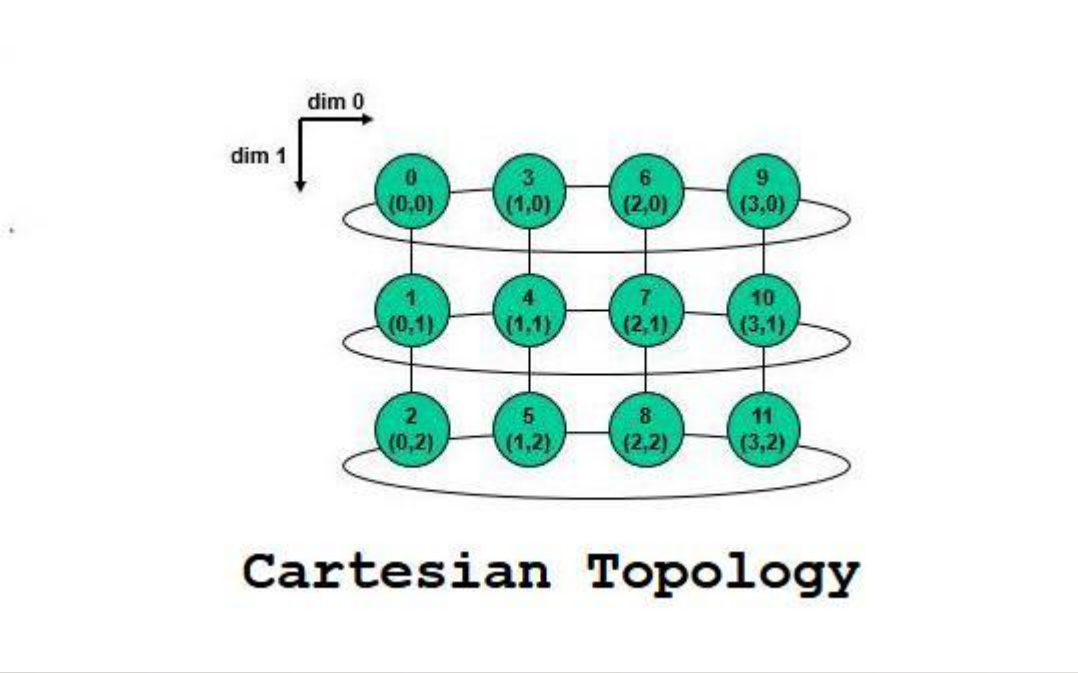
n > N



Процессоры нетрадиционной архитектуры

Примеры топологий





Почему необходимо использовать разные топологии параллельной передачи данных?

Топология должна соответствовать структуре вычислительной задачи
Тогда накладные расходы на связь уменьшаются.
Например: Топология 2D-матрицы удобна для умножения двух матриц.
Топология звезды удобна для задач ведущий-ведомый

Процессоры нетрадиционной архитектуры

Высокая стоимость первых параллельных систем:

- процессоры нетрадиционной архитектуры
- специальные языки программирования

Например: Transputer (1989)

Первый язык программирования для Transputer — OCCAM:

- нативный, императивный процедурный язык программирования

Параллельные компьютеры

Современные принципы построения параллельных систем:

- Стандартное оборудование (традиционная архитектура) Параллельные компьютеры могут быть построены из дешевых, широко распространенных компонентов.
- Стандартное серийное ПО (C, C++,..) + специальная библиотека с большим набором функций для:
  - ✔ создания связей между параллельными процессами
  - ✔ передачи сообщений между параллельными процессами

Параллельные вычисления

For example:

OpenMP (**Open Multi-Processing**) - C, C++ and Fortran http://openmp.org (Symmetric Malty Processor)

Message Passing Interface (**MPI**) - C, C++ and Fortran

http://www.mpi-forum.org (Massive Parallel Processing)

Parallel Virtual Machine (**PVM**) - software tool for parallel networking of computers - C, C++ and Fortran

http://www.csm.ornl.gov/pvm

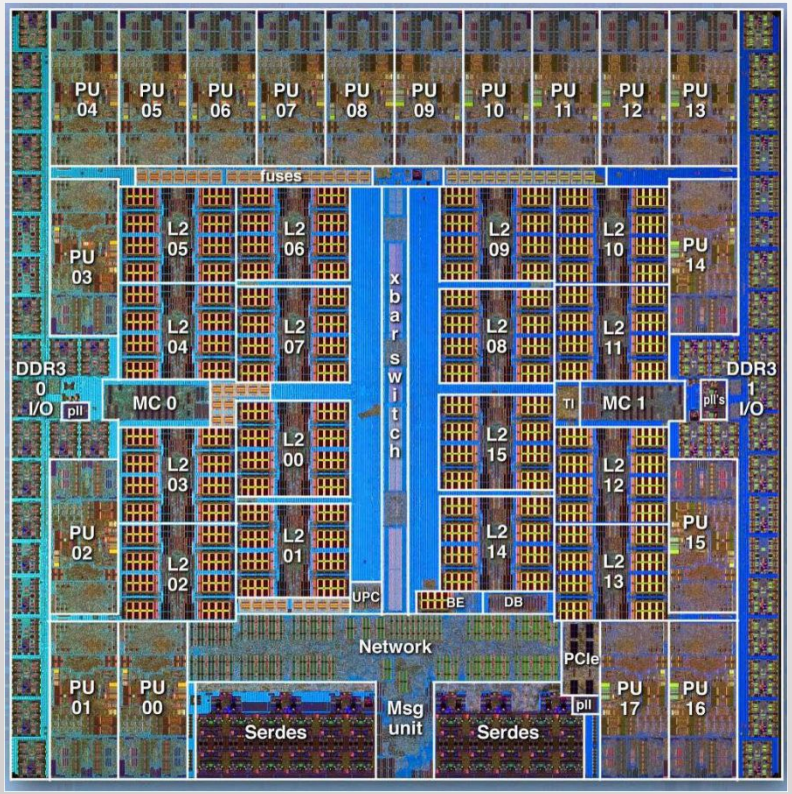
Эти системы являются стандартными и могут быть установлены на стандартных компьютерах, в сетях и кластерах.

Параллельные компьютеры

1. Все компьютеры сегодня параллельны по аппаратному обеспечению, т.к. содержат:

- несколько исполнительных блоков/ядер - PU
- Несколько функциональных блоков (кэш L1, кэш L2 и т. д.)
- несколько аппаратных потоков
- блоки преобразования данных из параллельного представления в последовательное и обратно -SerDes

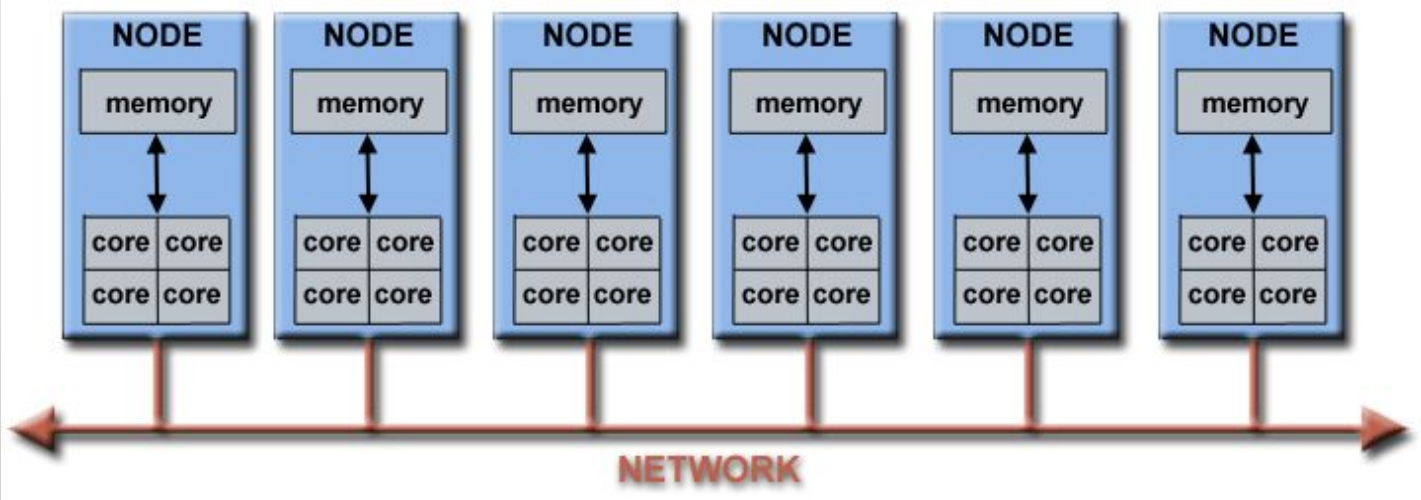
SMP - Симметричная многопроцессорная обработка или многопроцессорная обработка с общей памятью



IBM BG/Q Compute Chip with 18 cores (PU) and 16 L2 Cache units (L2)

Параллельные компьютеры

2. Сети соединяют несколько автономных компьютеров (узлов), чтобы сделать более крупный параллельный компьютер **clusters (MPP)**

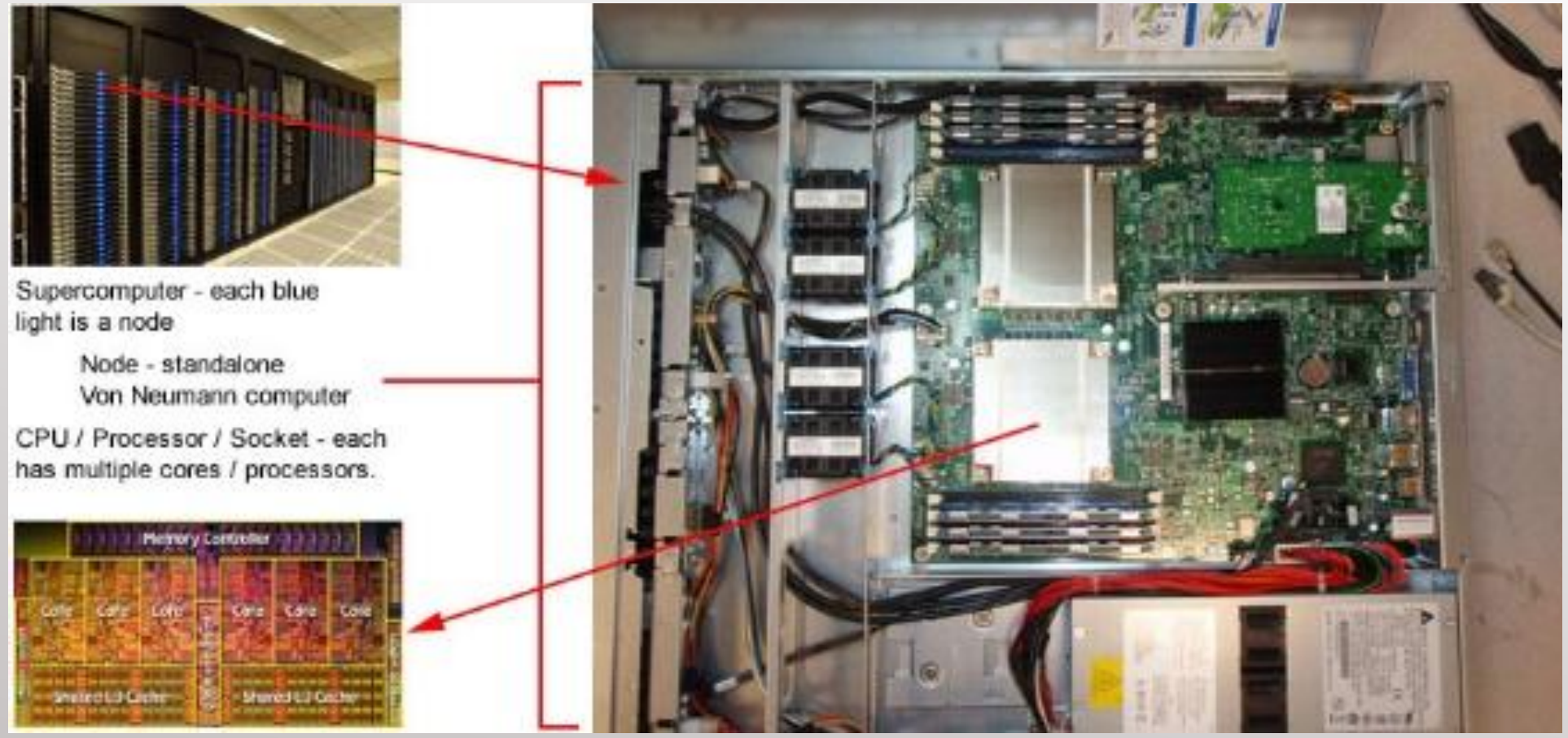


- Каждый вычислительный узел представляет собой многопроцессорный параллельный компьютер.
- Несколько вычислительных узлов объединены в сеть

Чем отличаются: Distributed / Parallel computations

Параллельные компьютеры

Example: modern supercomputer



- All components are placed next to each other.
- network is superfast (up to 1,6 Terabit/s)

введение
Что такое параллельные вычисления?

Зачем использовать параллельные вычисления?

- Почему у нас есть ограничения для последовательных вычислений?

- Какие причины создают существенные ограничения для простого создания более быстрых последовательных компьютеров?

Зачем использовать параллельные вычисления?

Ограничения для последовательных вычислений.

А) Физические причины:

- Скорость передачи (Baud rate) сигнала:
  - ограничение пропускания медного провода (9 см/нс)
  - ограничивает скорость света (30 см/наносекунда)
- Ограничения миниатюризации - современные технологии позволяют размещать на кристалле все больше и больше транзисторов. Но предельный размер компонентов определяется на молекулярном или атомном уровне.

Зачем использовать параллельные вычисления?

Б) Экономические ограничения:

- Все дороже сделать один сверхбыстрый процессор.
- Производство нескольких более умеренно быстрых процессоров в одном чипе:
  - Дают такую же (или лучшую) производительностьесть
  - Дешевле

Современные компьютеры используют аппаратный параллелизм для повышения производительности.

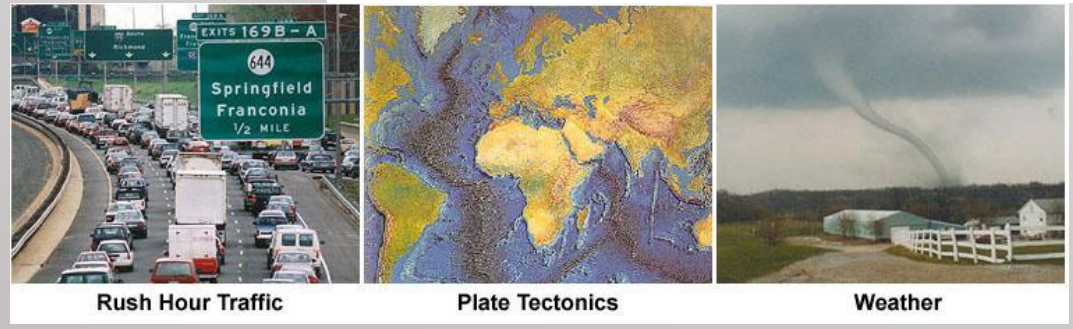
ВВЕДЕНИЕ
Где используются параллельные вычисления? примеры

Зачем использовать параллельные вычисления?

Основные причины:

1. Реальный мир массивно параллелен:

Параллельные вычисления гораздо лучше подходят для моделирования, симуляции и понимания сложных явлений реального **мира**.



Зачем использовать параллельные вычисления?

2. Экономия времени и/или денег:

- Сокращение времени выполнения задачи с потенциальной экономией средств. (Пример: режим реального времени)

- Параллельные компьютеры можно собрать из дешевых, массовых компонентов.

Зачем использовать параллельные вычисления?

3. Решение больших сложных проблемм:

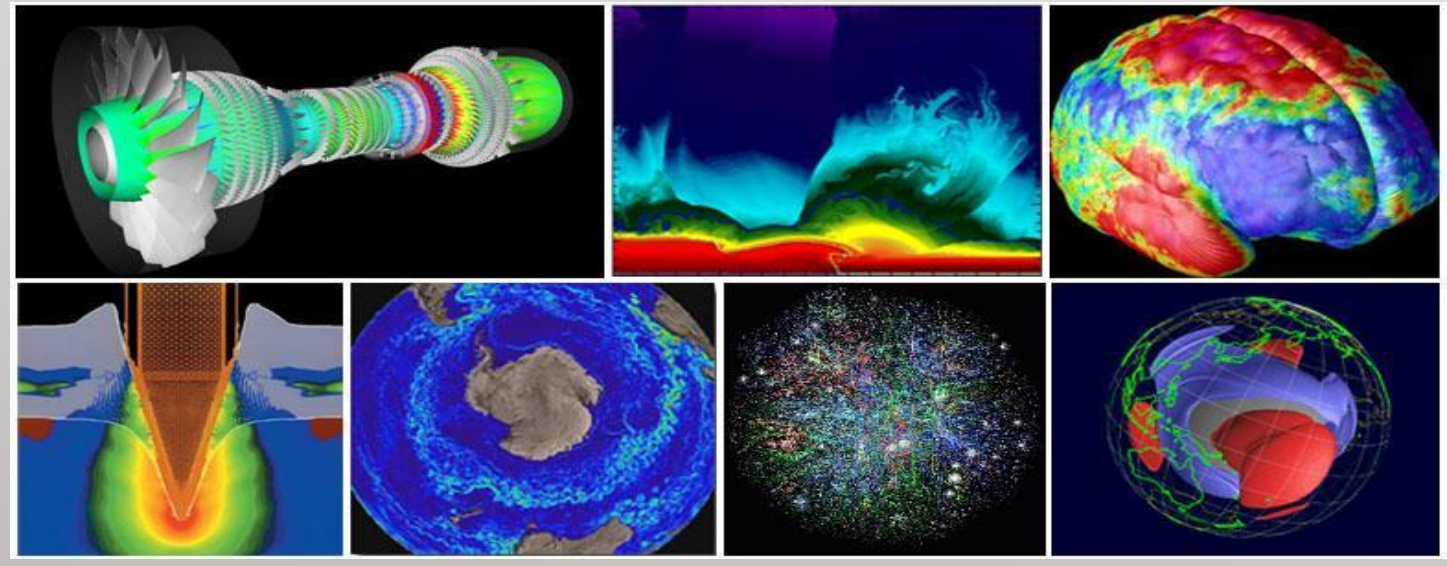
Многие задачи настолько велики и/или сложны, что их невозможно решить на одном компьютере (особенно при ограниченной памяти компьютера).

Например:

а) «Большие задачи-вызовы» (en.wikipedia.org/wiki/Grand_Challenge), требующие петафлопсов и петабайт вычислительных ресурсов.

б) Поисковые системы

(базы данных, которые обрабатывают миллионы транзакций в секунду)



Why Use Parallel Computing?

For reference only:

Computer performance	
Name	FLOPS
yottaFLOPS	10^{24}
zettaFLOPS	10^{21}
exaFLOPS	10^{18}
petaFLOPS	10^{15}
teraFLOPS	10^{12}
gigaFLOPS	10^{9}
megaFLOPS	10^{6}

Зачем использовать параллельные вычисления?

4. Поддержка параллельного взаимодействия:

Например: совместные сети обеспечивают глобальное общее место, где люди со всего мира могут встречаться и работать «виртуально».(https://web.archive.org/web/20040627082707/http://www.accessgrid.org/)

Зачем использовать параллельные вычисления?

5. Совместное использование использование вычислительных ресурсов в глобальной сети.

Example 1:

- SETI@home (setiathome.berkeley.edu)

SETI@home — это научный эксперимент, проводимый Калифорнийским университетом в Беркли, в котором компьютеры, подключенные к Интернету, используются для поиска внеземного разума (SETI). Вы можете принять участие, запустив бесплатную программу, которая загружает и анализирует данные радиотелескопа.

Зачем использовать параллельные вычисления?

Совместное использование использование вычислительных ресурсов в глобальной сети.

Example 2:

- **Folding@home** (folding.stanford.edu)

«Folding@home — проект, ориентированный на исследование болезней. «Проблемы, которые мы решаем, требуют очень больших компьютерных вычислений — и нам нужна ваша помощь, чтобы найти лекарства!»

ВМЕСТЕ МЫ СИЛА

Введение Области использлвания параллельных вычислений?

Кто использует параллельные вычисления?

- **Наука и техника**

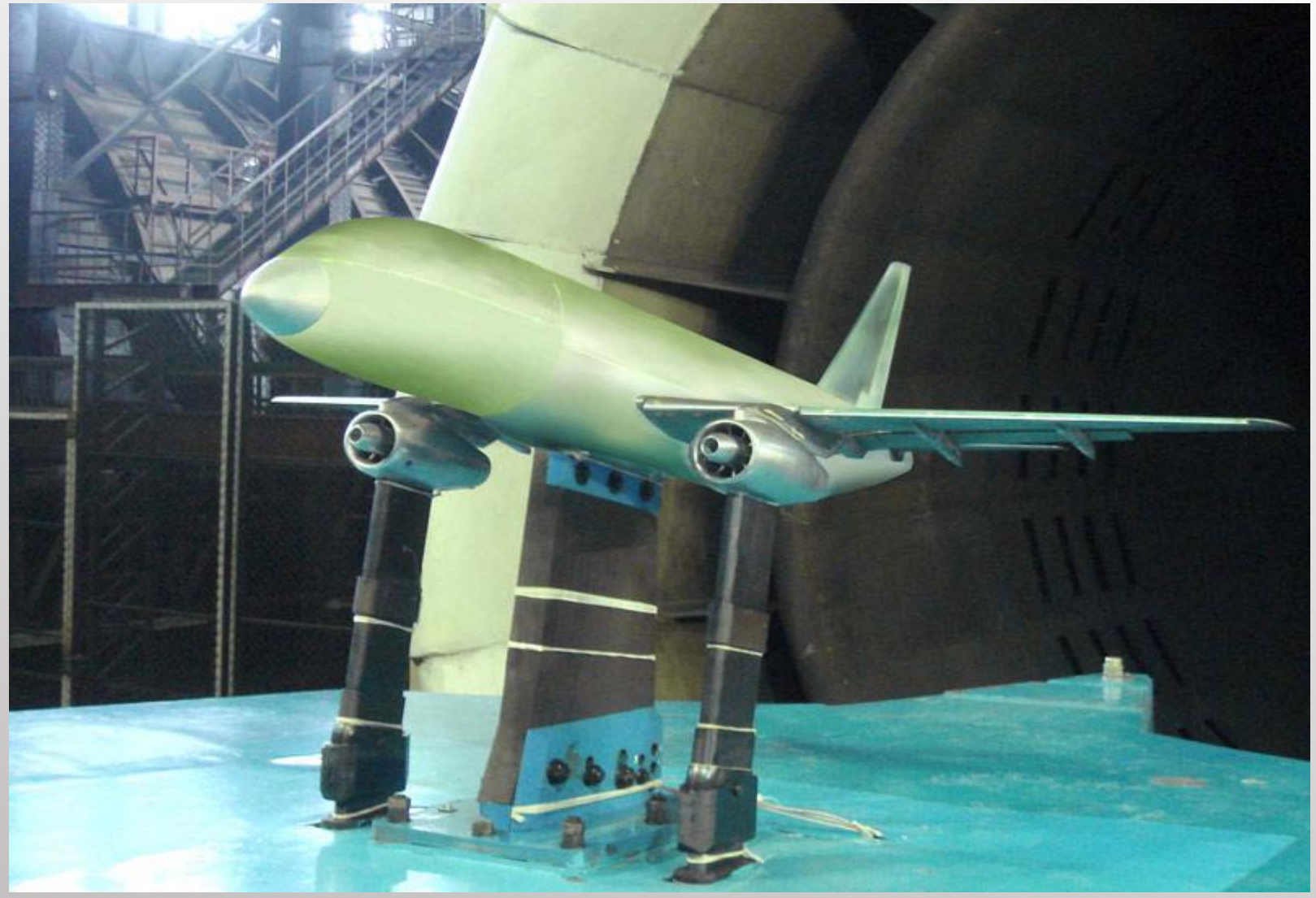
- Атмосфера, Земля, Окружающая среда
- Физика - прикладная, ядерная, частица, конденсированное состояние, высокое давление, синтез, фотоника
- Бионаука, Биотехнология, Генетика
- Химия, Молекулярные науки
- Геология, Сейсмология
- Машиностроение - от протезов до космических кораблей
- Электротехника, Схемотехника, Микроэлектроника
- Информатика, Математика
- Защита, Оружие

Кто использует параллельные вычисления?

- **Промышленность и коммерция** - движущие силы в развитии более быстрых компьютеров:

- «Большие данные», базы данных, интеллектуальный анализ данных
- Искусственный интеллект
- Поисковые системы в Интернете, бизнес-услуги в Интернете
- Медицинская визуализация и диагностика
- Фармацевтический дизайн
- Финансово-экономическое моделирование
- Управление национальными и транснациональными корпорациями
- Передовая графика и виртуальная реальность, особенно в индустрии развлечений
- Сетевое видео и мультимедийные технологии
- Разведка нефти

Пример: аэродинамическая труба (Wind tunnel)

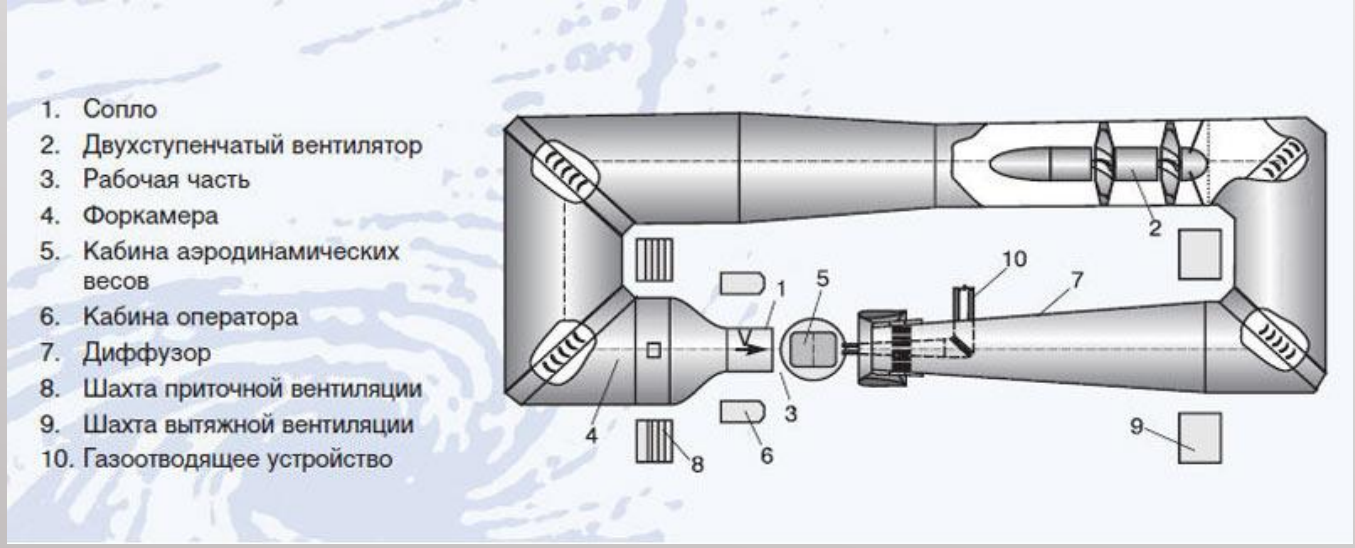


Аэродинамическая труба(Wind tunnel) Т-104, ЦАГИ

Центральный аэрогидродинамический институт имени профессора Н. Е. Жуковского

- Скорость потока - 10–120 м/с
- Диаметр сопла- 7 м Длина рабочей части - 13 м
- Мощность вентилятора - **28.4 МВт**

http://www.tsagi.ru/experimental_base/aerodinamicheskaya-truba-t-104/



Supercomputer СКИФ МГУ

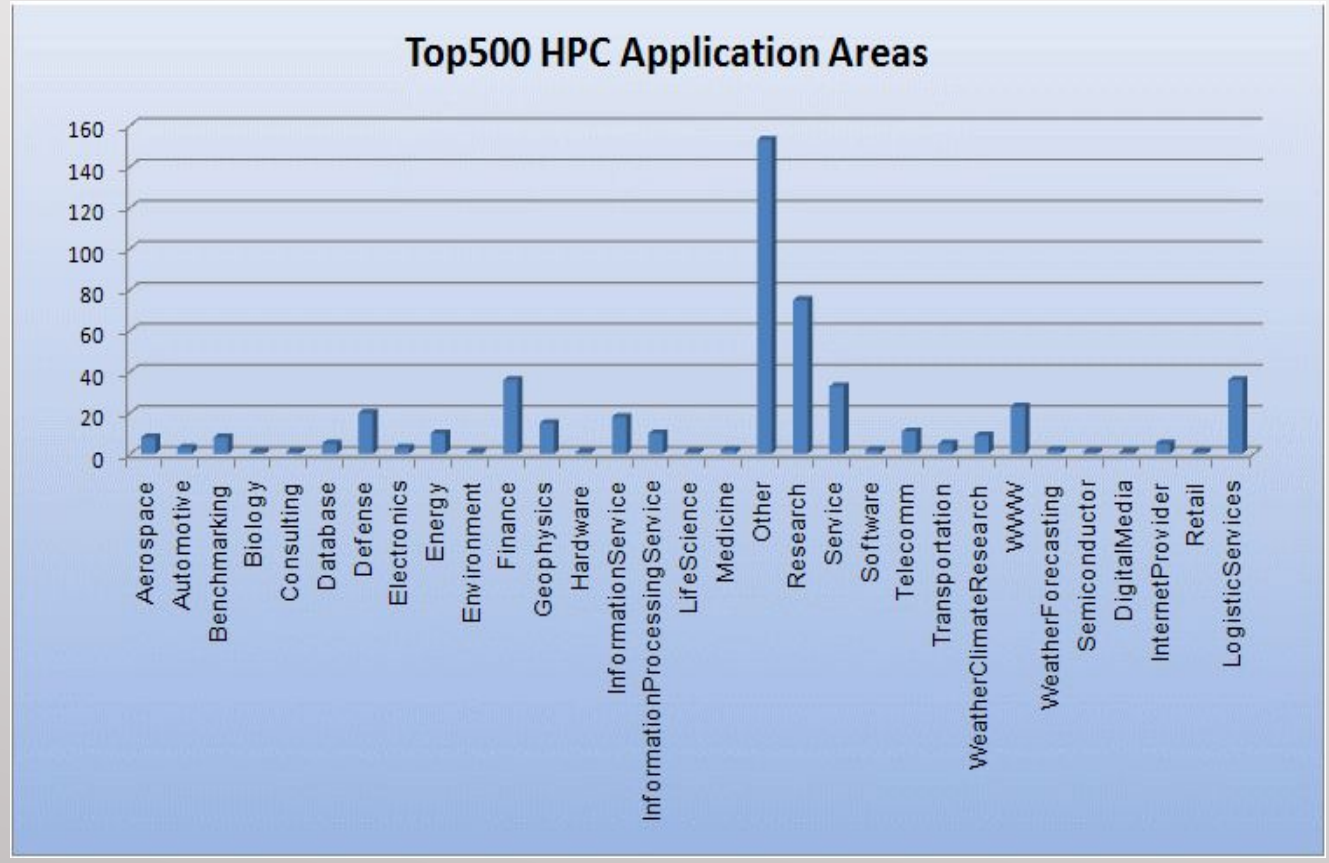
Аэродинамическая труба числовая

- Пиковая производительность - 60 TFlop/s (10^{12})
- До ноября 2011 года входил в список Топ-500 самых мощных компьютеров мира. Потребляемая мощность - **0.72 МВт**
  http://parallel.ru/cluster/skif_msu.html

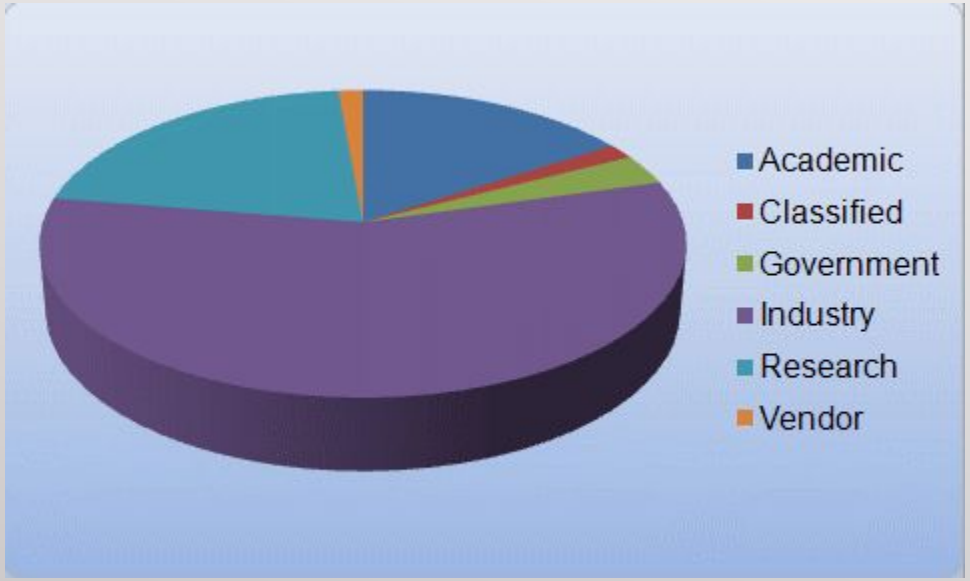
Lomonosov Moscow State University

Кто использует параллельные вычисления?

- Области применения



Кто использует параллельные вычисления?



Понятия и терминология

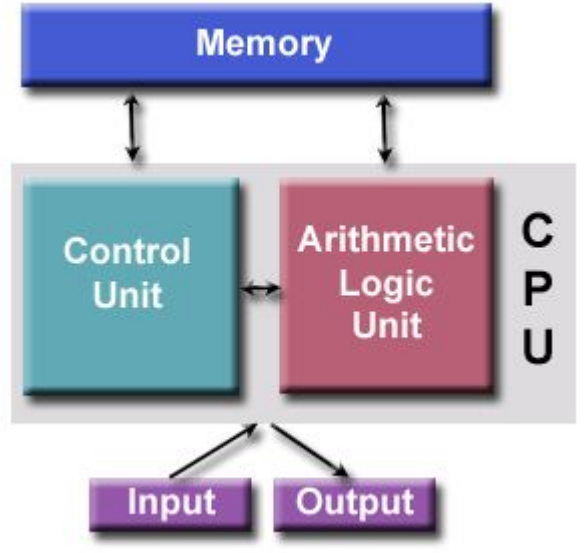
Архитектура фон Неймана

Венгерский математик/гений

Джон фон Нейман около 1940-х (Источник: архивы LANL)

С 1945 года все компьютеры имеют эту базовую конструкцию!

Архитектура фон Неймана



- Четыре основных компонента: память, блок управления, арифметико-логическое устройство, ввод/вывод.

- Оперативная память для хранения программных инструкций и данных

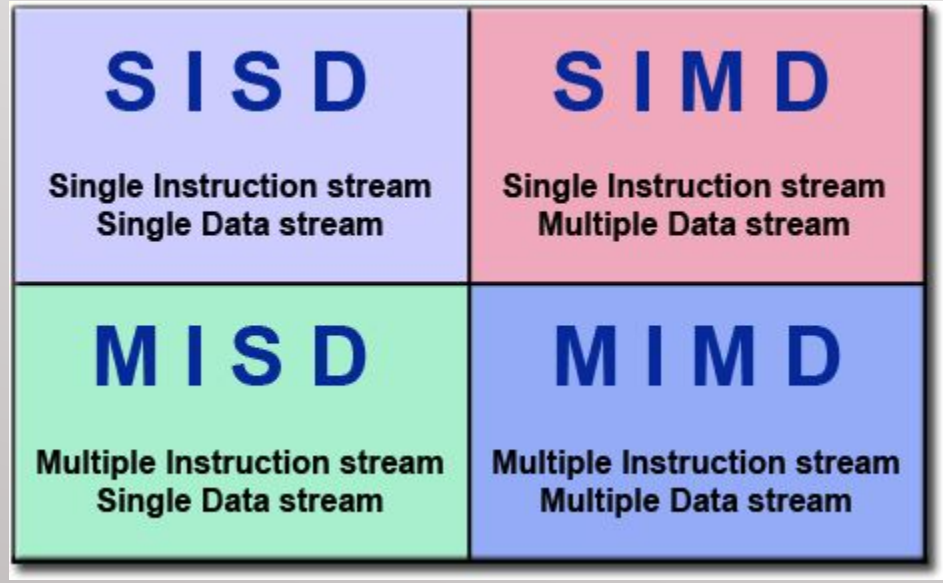
-Блок управления: берет инструкции/данные из памяти и последовательно выполняет инструкции.
-Арифметический блок: выполняет основные арифметические операции
- Ввод/вывод - это интерфейс для человека-оператора.

Классификация Флинна (Flynn's Classical Taxonomy)

Таксономия Флинна — широко используемая классификация с 1966 года.

Таксономия Флинна определяет архитектуру многопроцессорных компьютеров в двух измерениях:

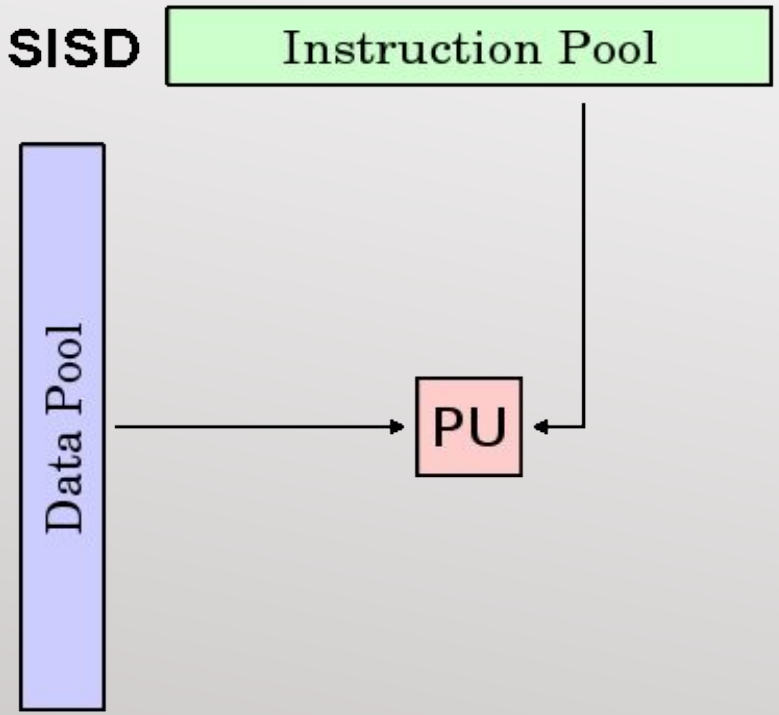
- Поток инструкций и поток данных.
- Каждый из потоков может быть: одиночным или множественным



Классификация Флинна

Single Instruction, Single Data (SISD):

- Последовательный (непараллельный) компьютер
- Архитектура фон Неймана
- **Детерминированное** исполнение -?



Детерминизм

Детерминированные компьютерные программы всегда будет давать один и тот же результат с одним и тем же набором входных данных.

Недетерминированные компьютерные программы не всегда будут давать один и тот же результат при одном и том же наборе входных данных.

Классификация Флинна

Single Instruction, Single Data (SISD):

- Это самый старый тип компьютера

UNIVAC1

IBM 360

CRAY1

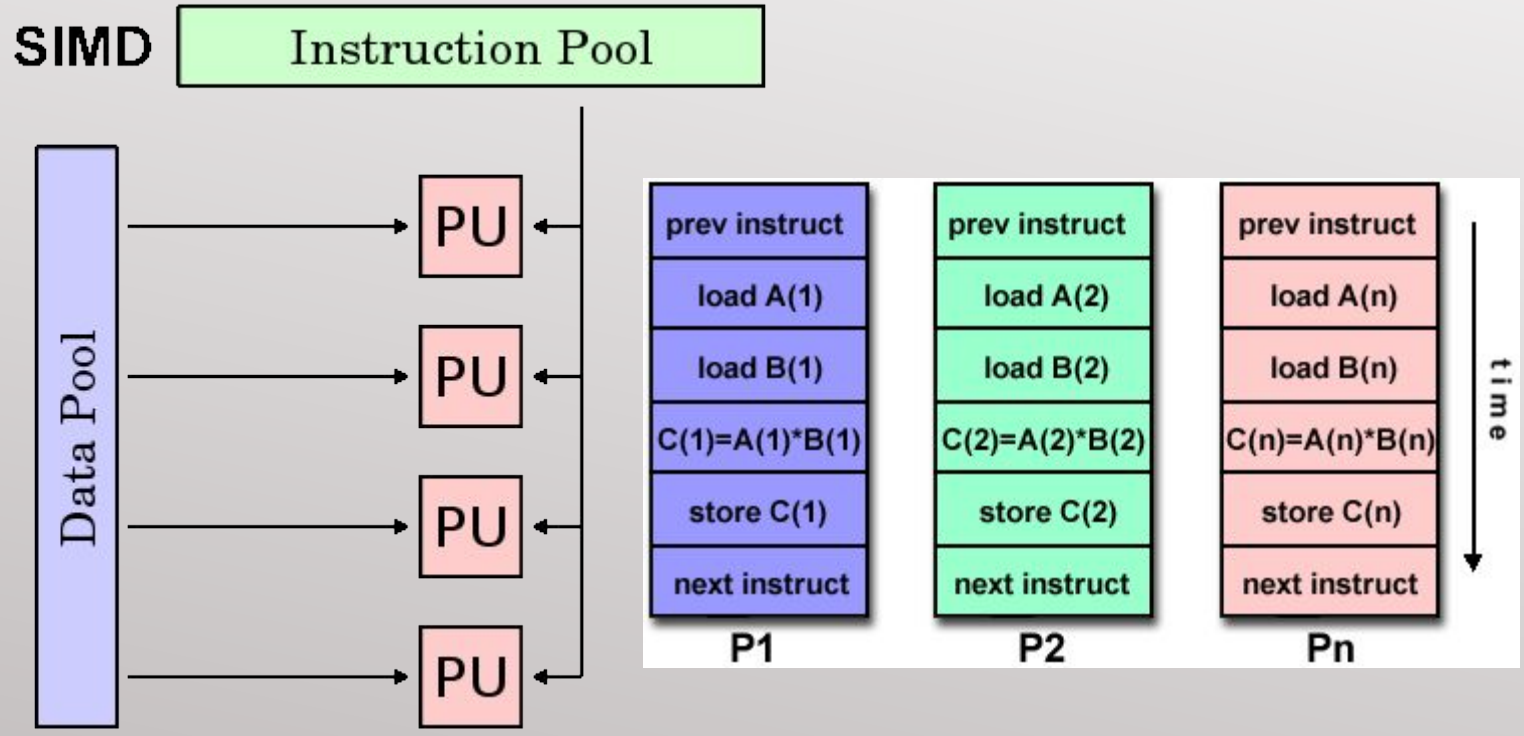
CDC 7600

PDP1

Dell Laptop

Классификация Флинна

Single Instruction, Multiple Data (SIMD)



Flynn's Classical Taxonomy

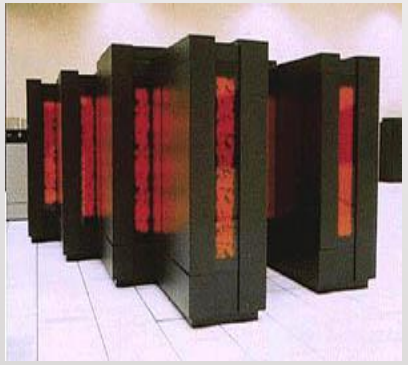
Single Instruction, Multiple Data (SIMD):

- Тип параллельного компьютера
- Используется для специализированных задач, характеризующихся высокой степенью регулярности, таких как обработка графики/изображений.
- Детерминированное исполнение

Классификация Флинна

Single Instruction, Multiple Data (SIMD) –конвеерно - векторные суперкомпьютеры:

MasPar



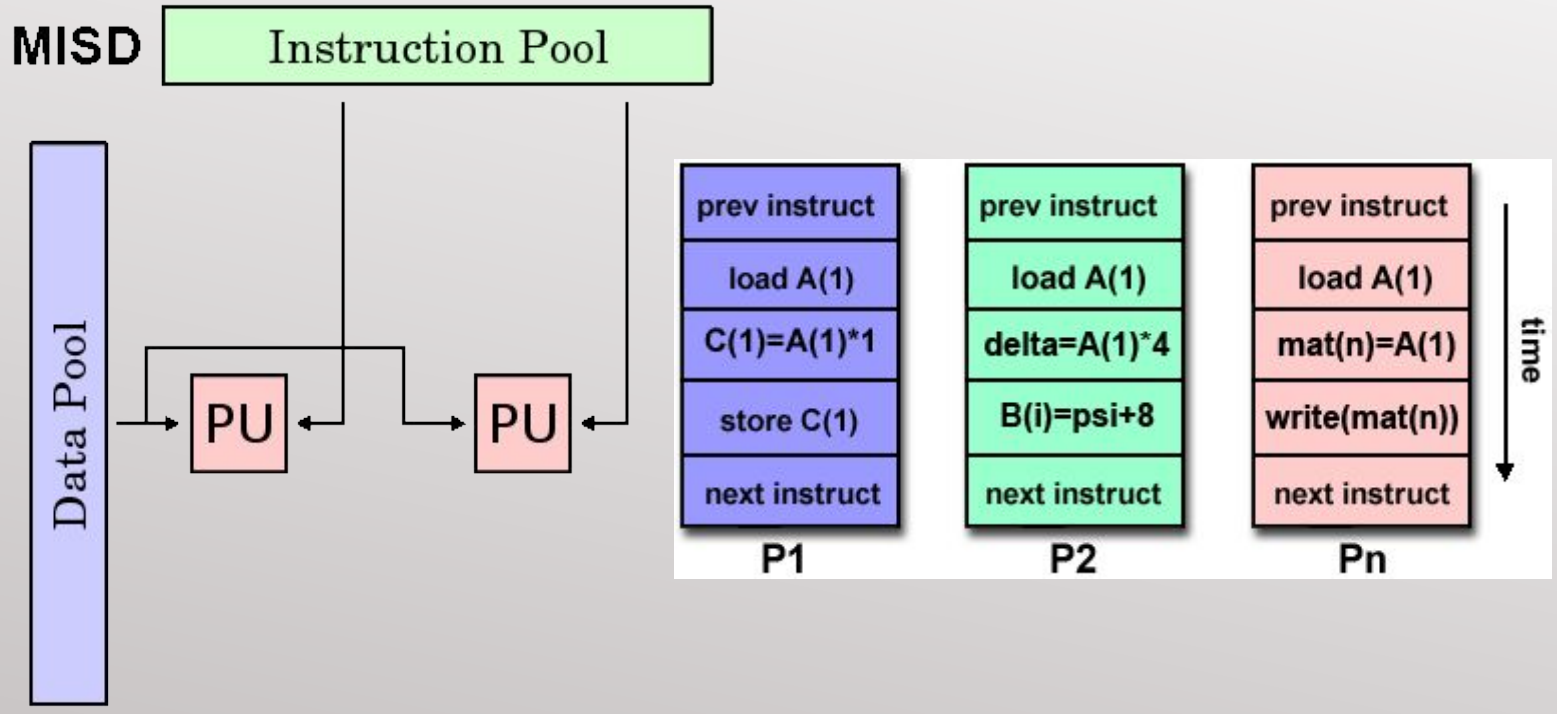
Thinking Machines CM-2

Cray X-MP

Cray Y-MP

Классификация Флинна

Multiple Instruction, Single Data (MISD):



Классификация Флинна

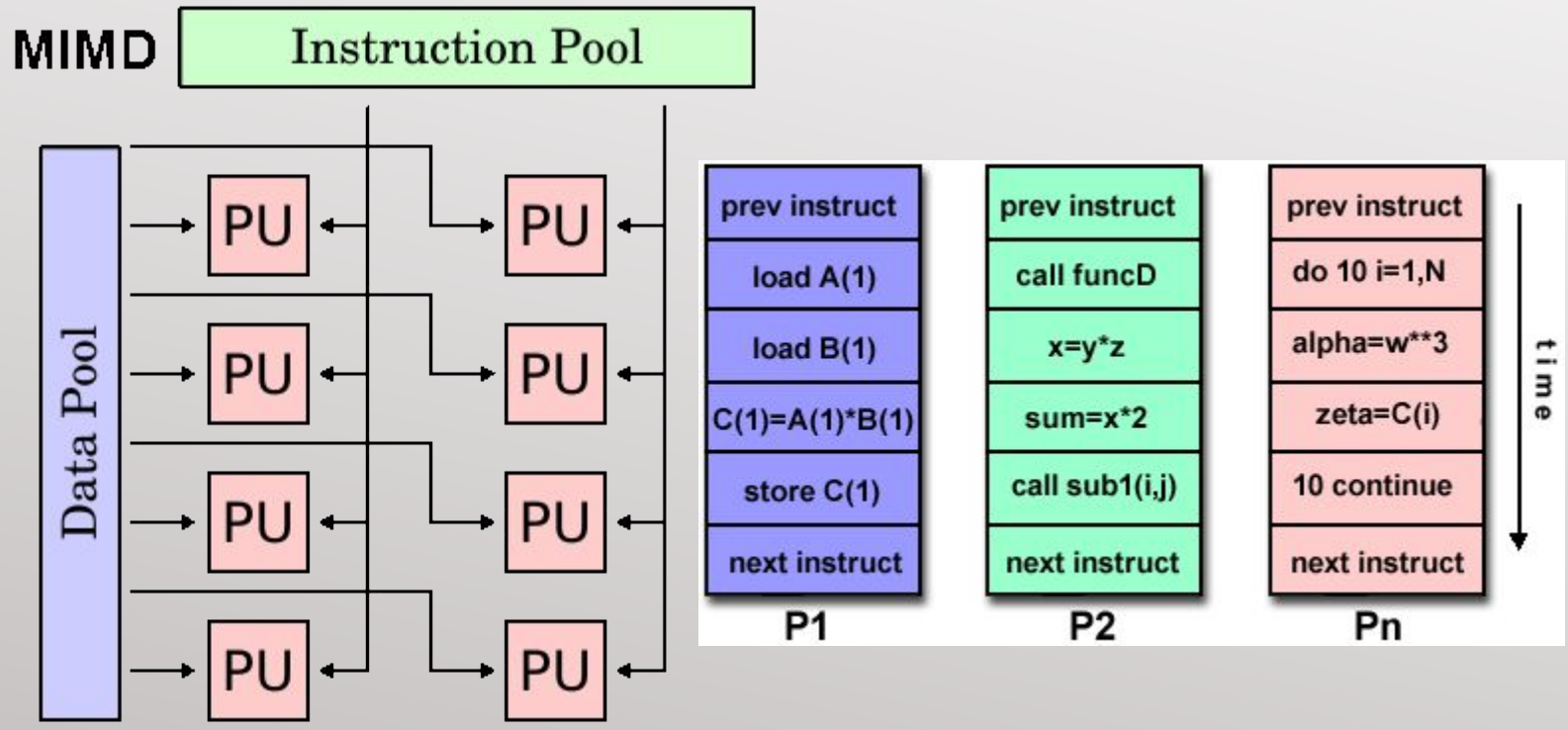
Multiple Instruction, Single Data (MISD):

- Ттип параллельного компьютера
- Детерминированное исполнение
- Examples:

Существовало мало (если вообще было) реальных примеров этого класса параллельных компьютеров.

Классификация Флинна

Multiple Instruction, Multiple Data (MIMD):



Классификация Флинна

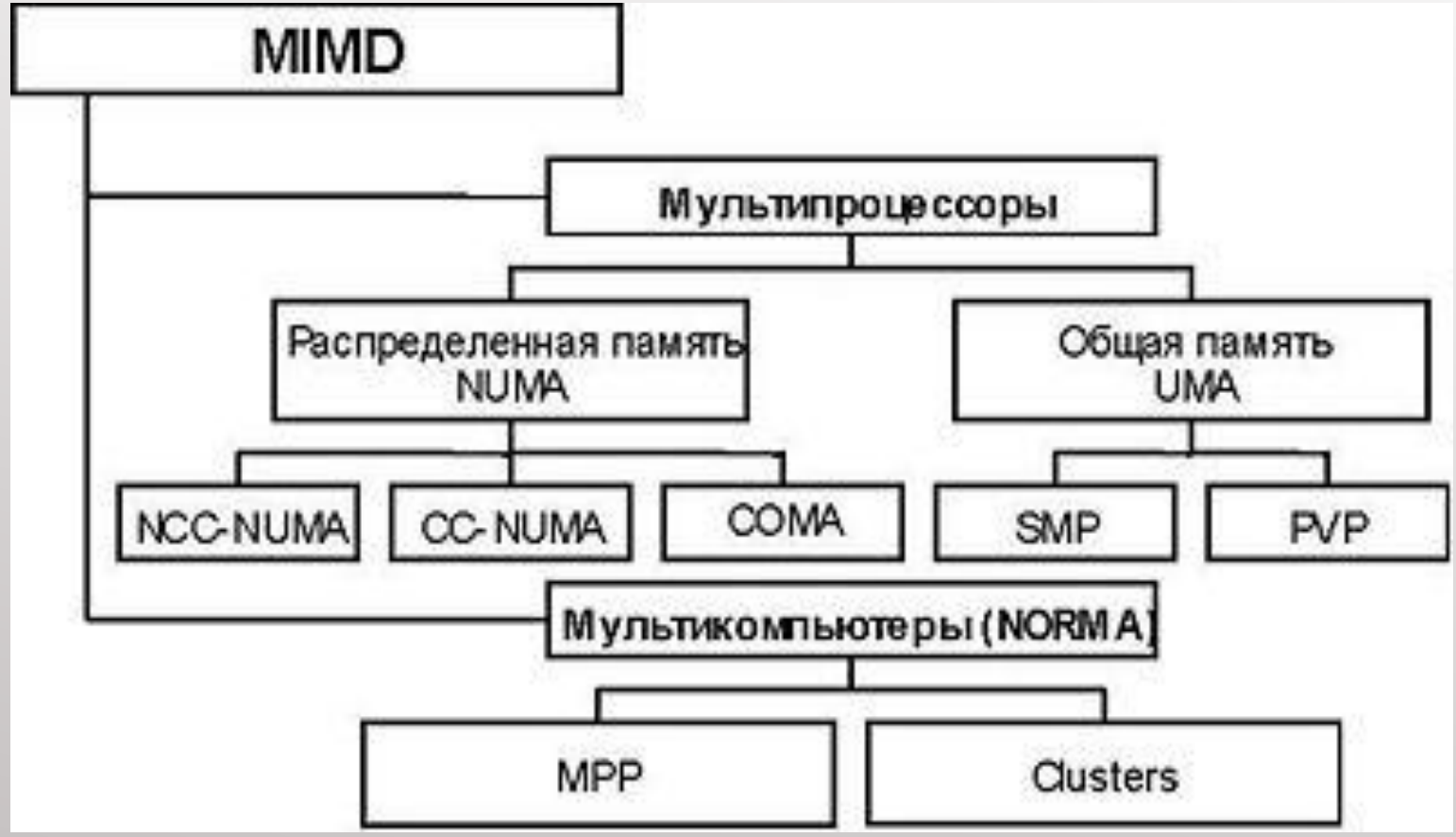
Multiple Instruction, Multiple Data (MIMD):

- Тип параллельного компьютера
- Выполнение может быть синхронным или асинхронным, **детерминированным** или **недетерминированным.**
- В настоящее время самый распространенный тип параллельных современных суперкомпьютеров

Примеры:

самые современные суперкомпьютеры, объединенные в сеть параллельные компьютерные кластеры и «сетки», многопроцессорные компьютеры SMP.

Классификация многопроцессорных ВС



Multiprocessors – мультипроцессоры

UMA (Uniform Memory Access) - **архитектура** памяти системы, в которой в которой время доступа всех процессоров не зависит от их расположения.

NUMA (**Non-Uniform Memory Access**) - **архитектура** памяти системы, в которой время доступа зависит от расположения памяти - доступ процессора к локальной памяти быстрее, чем к нелокальной.

NCC-NUMA (**Non Cache coherent NUMA**) – гибридная архитектура, в которой память физически распределена по различным частям системы, но логически разделена, так что пользователь видит единое адресное пространство.

CC-NUMA (**Cache coherent NUMA**) - система с кэш-когерентным доступом к неоднородной памяти.

COMA (**Cache Only Memory Architecture**)

SMP – (см. первую лекцию)

PVP (Parallel Vector Process) – параллельная архитектура с векторными процессорами