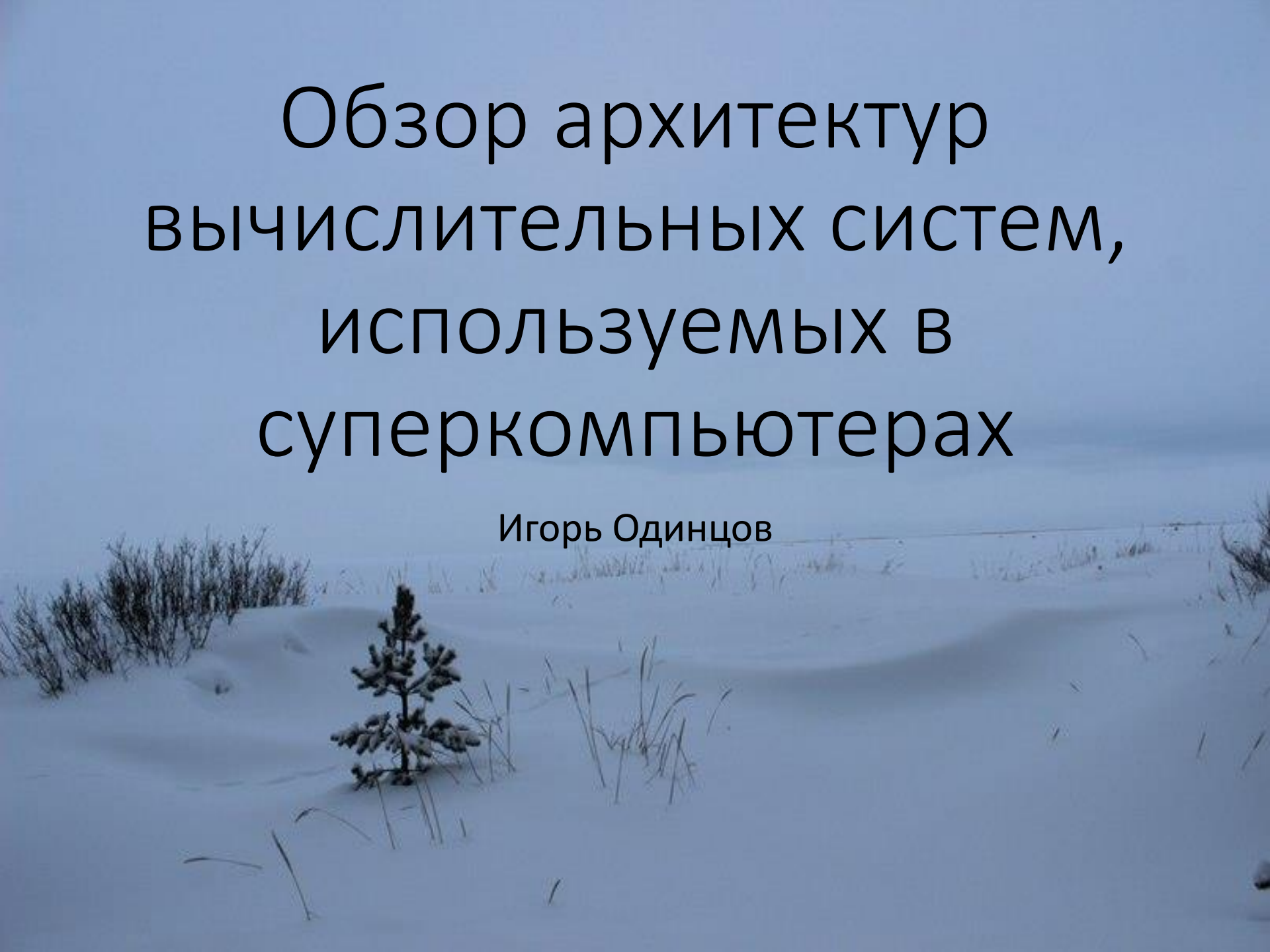
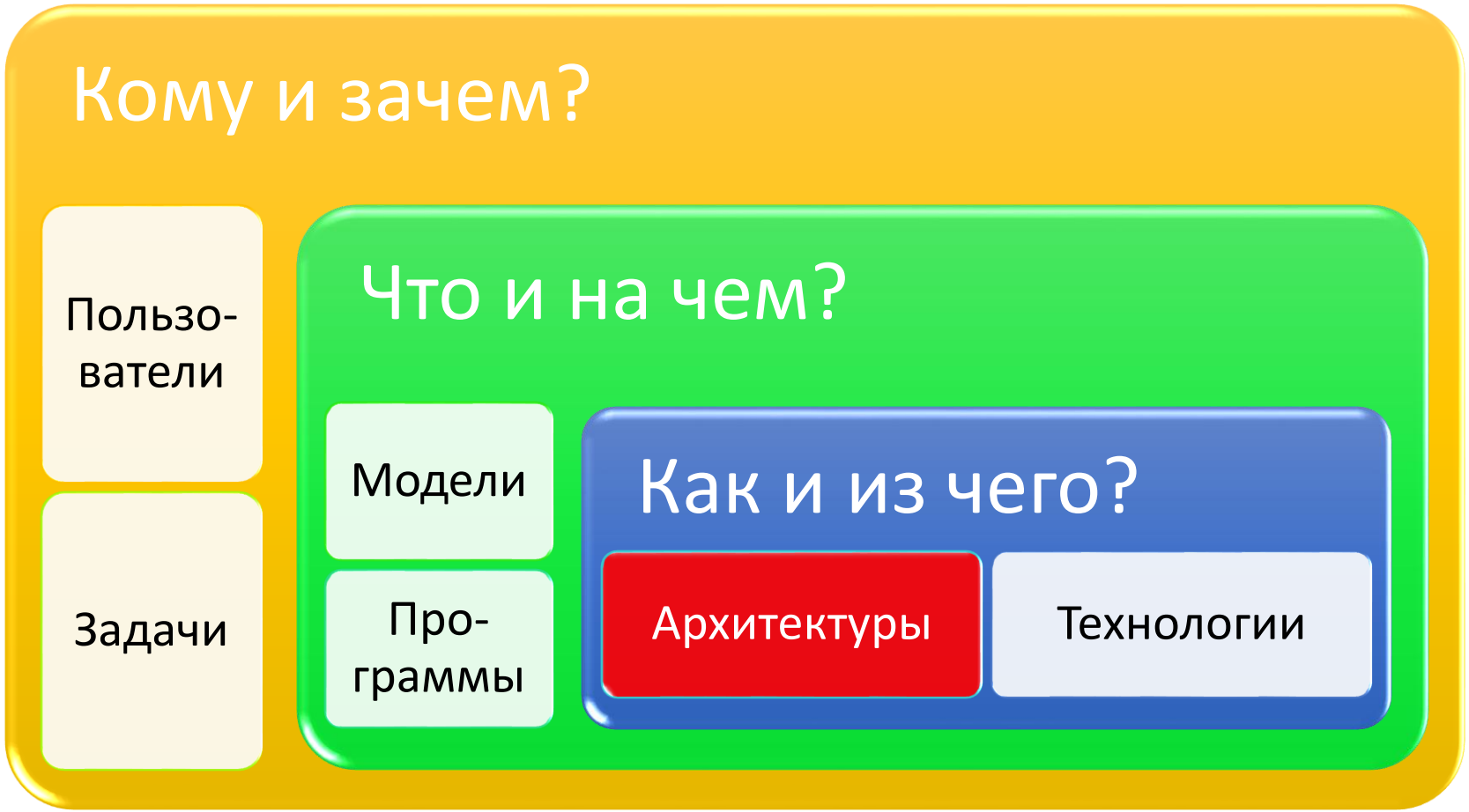
Обзор архитектур вычислительных систем, используемых в суперкомпьютерах

Игорь Одинцов



Архитектуры… И не только…

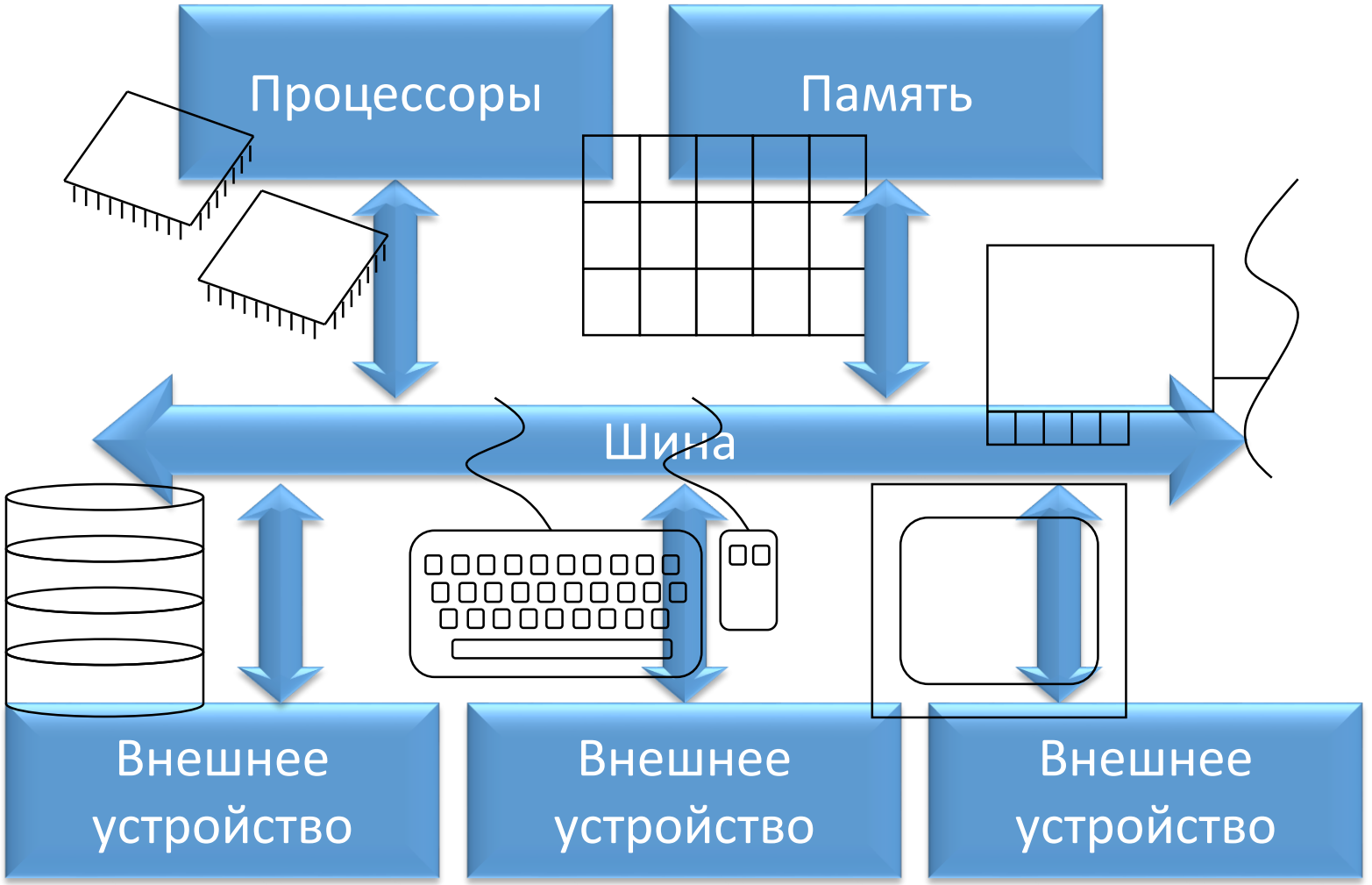


1. Технологии и архитектуры

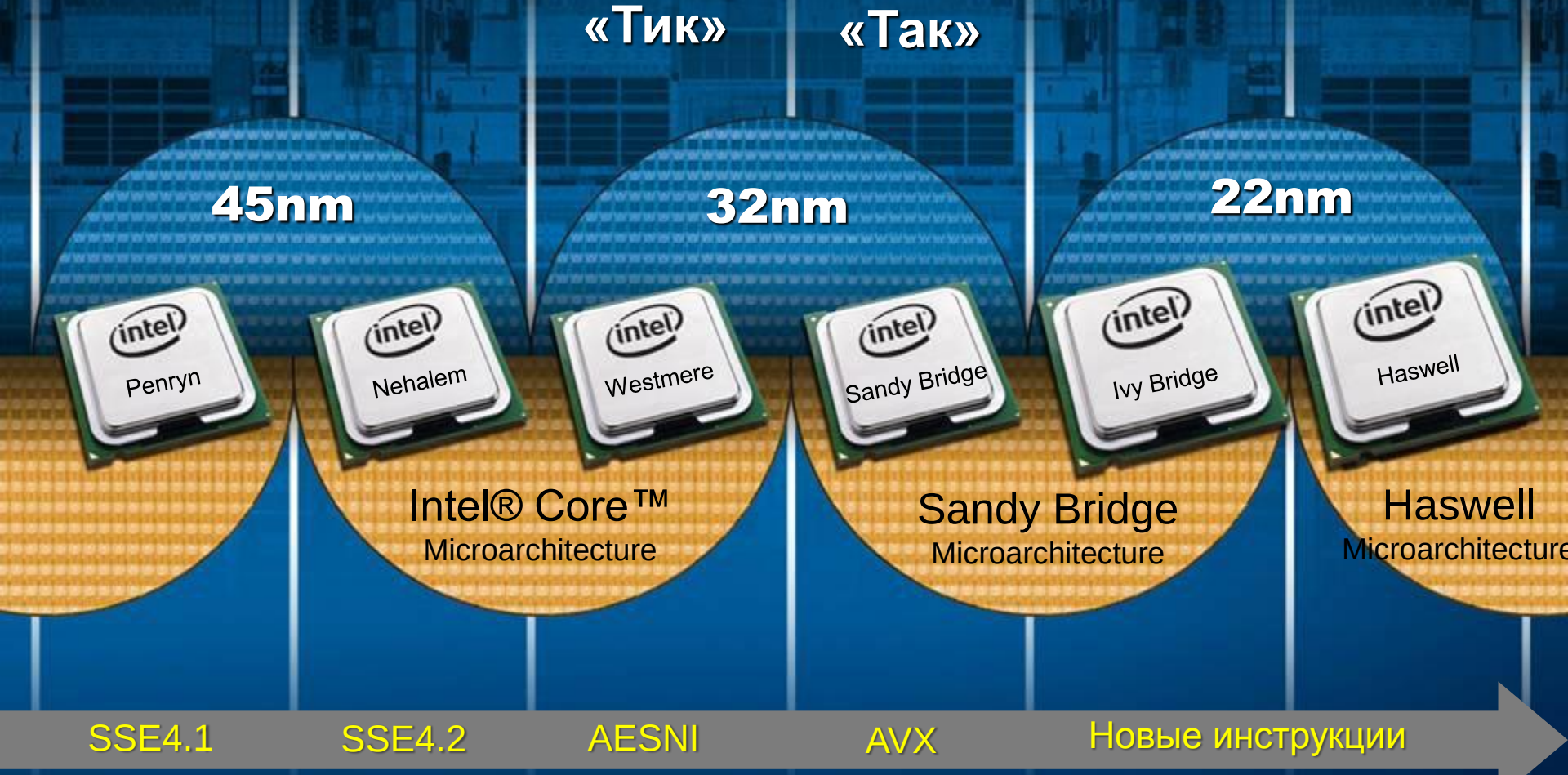
Как и из чего сделан (супер)компьютер?

А в чем разница между компьютером и суперкомпьютером?

Аппаратные ресурсы компьютера



Технологии и архитектуры

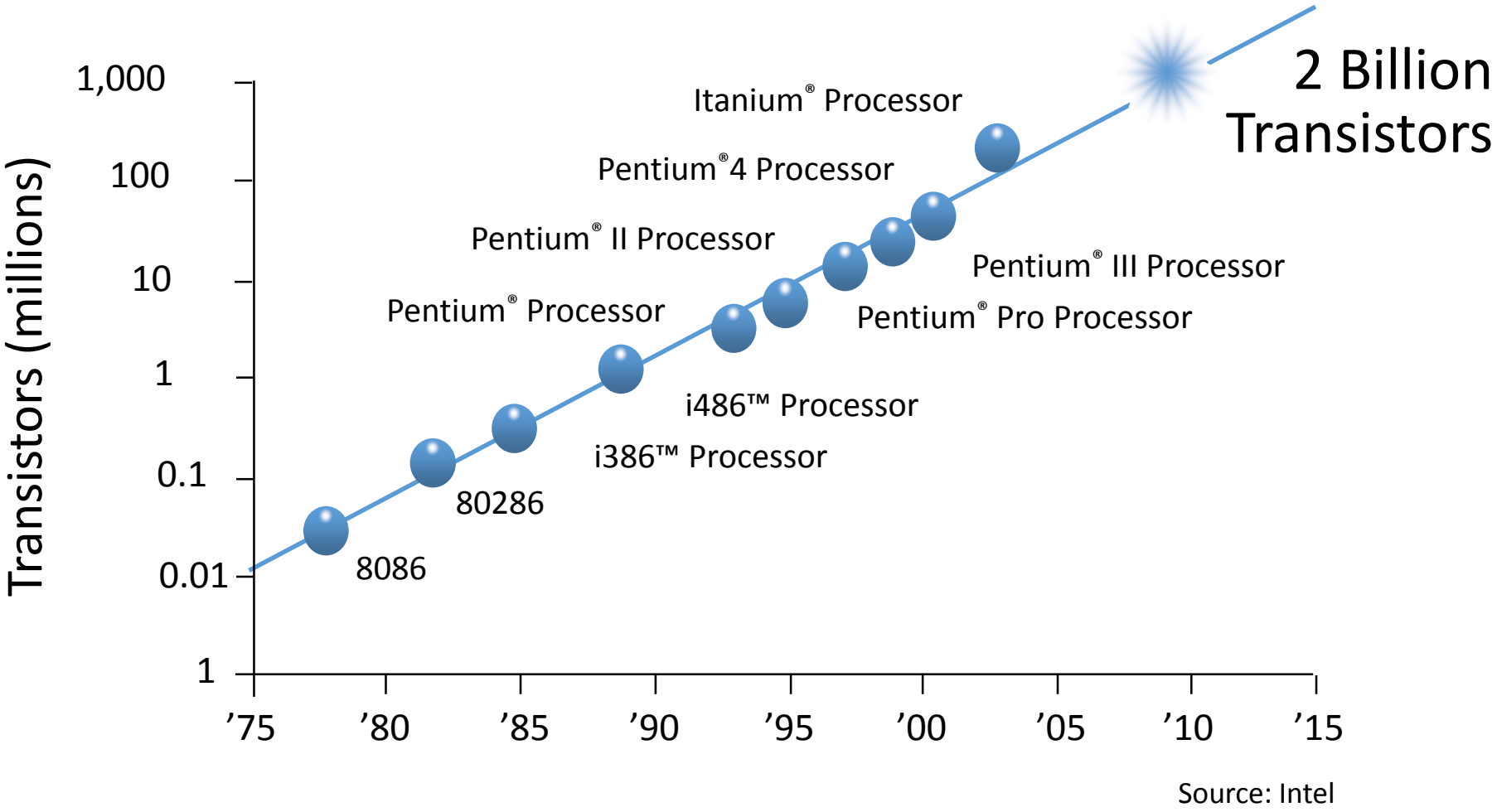


Potential future options, subject to change without notice.

Три важных темы при изучения технологии

- Закон Мура
- Нанометры
- Тактовая частота

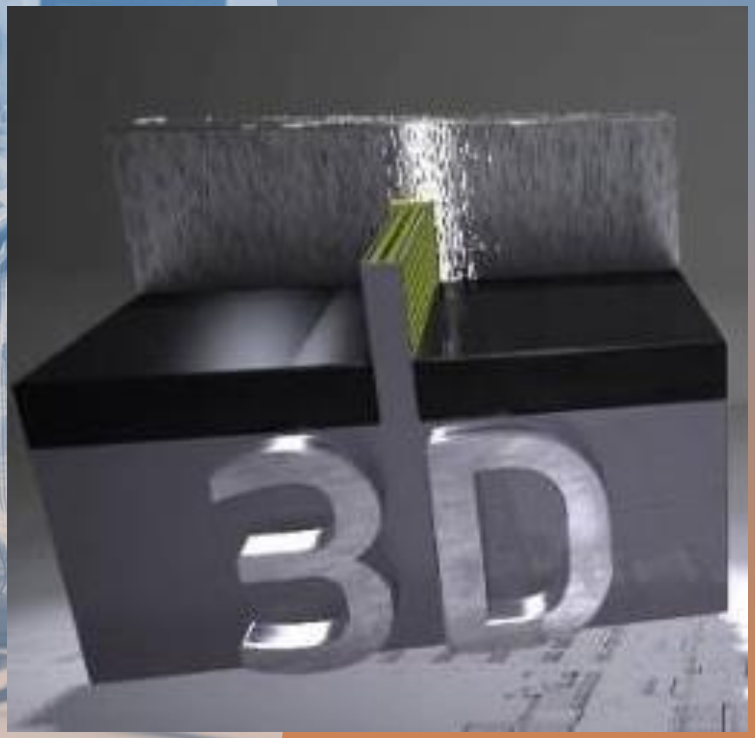
Закон Мура



Количество транзисторов на единице поверхности удваивается каждые 18 месяцев

Самый передовой технологический процесс

Продолжая новаторство
Реализуя закон Мура



Революционные 3-D Tri-Gate транзисторы Intel

- до 37% более высокая производительность
- до 50% снижение энергопотребления
- 22 и 14 нанометровый технический процесс
- снижение себестоимости производства

Беспрецедентная эффективность при значительном росте производительности

Фабрики

14nm

D1X – Oregon
Development Fab

Fab 42 – Arizona
High Volume Fab

22nm

D1D – Oregon

D1C – Oregon

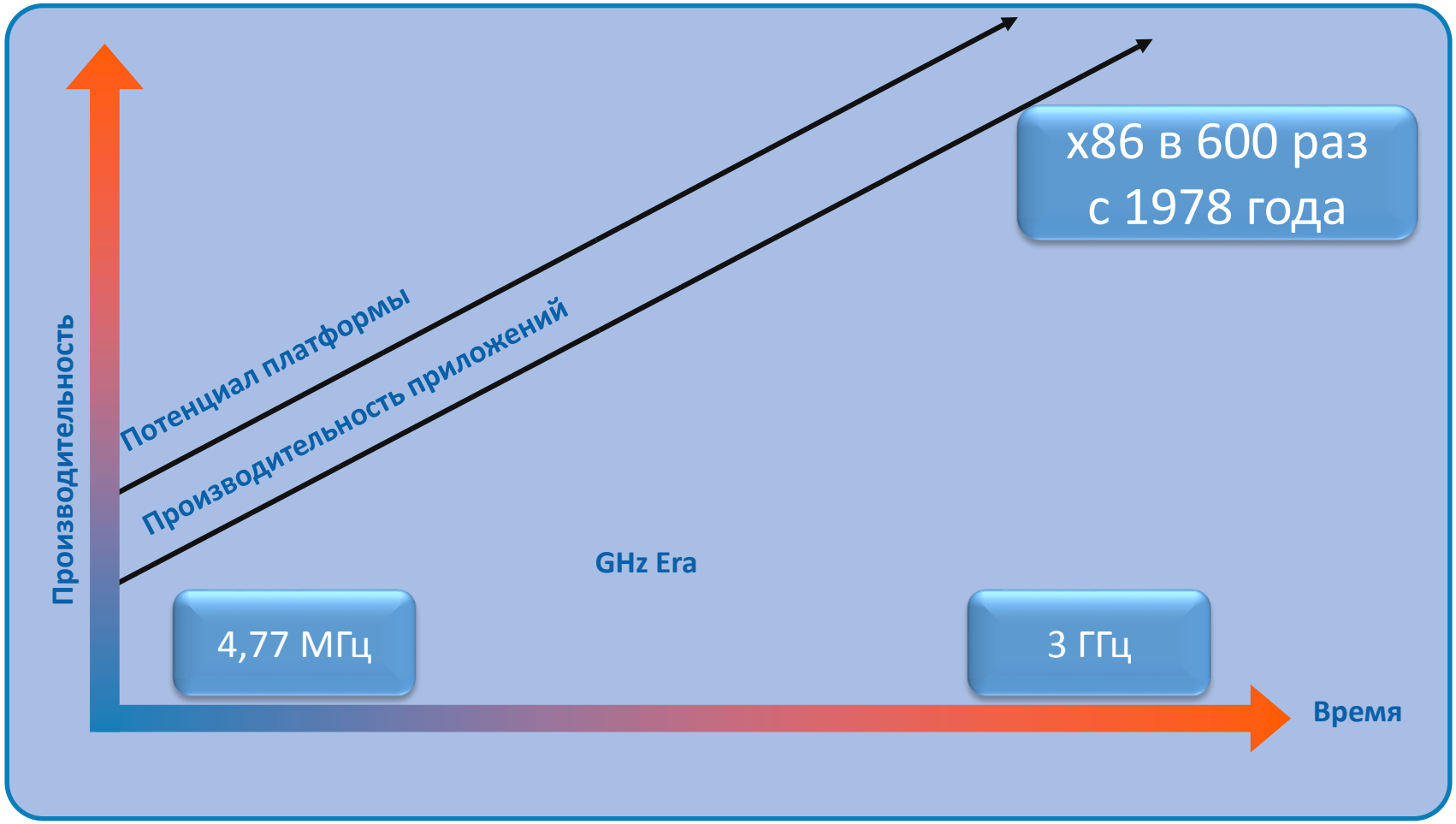
Fab 32 – Arizona

Fab 28 – Israel

Fab 12 – Arizona

Частота

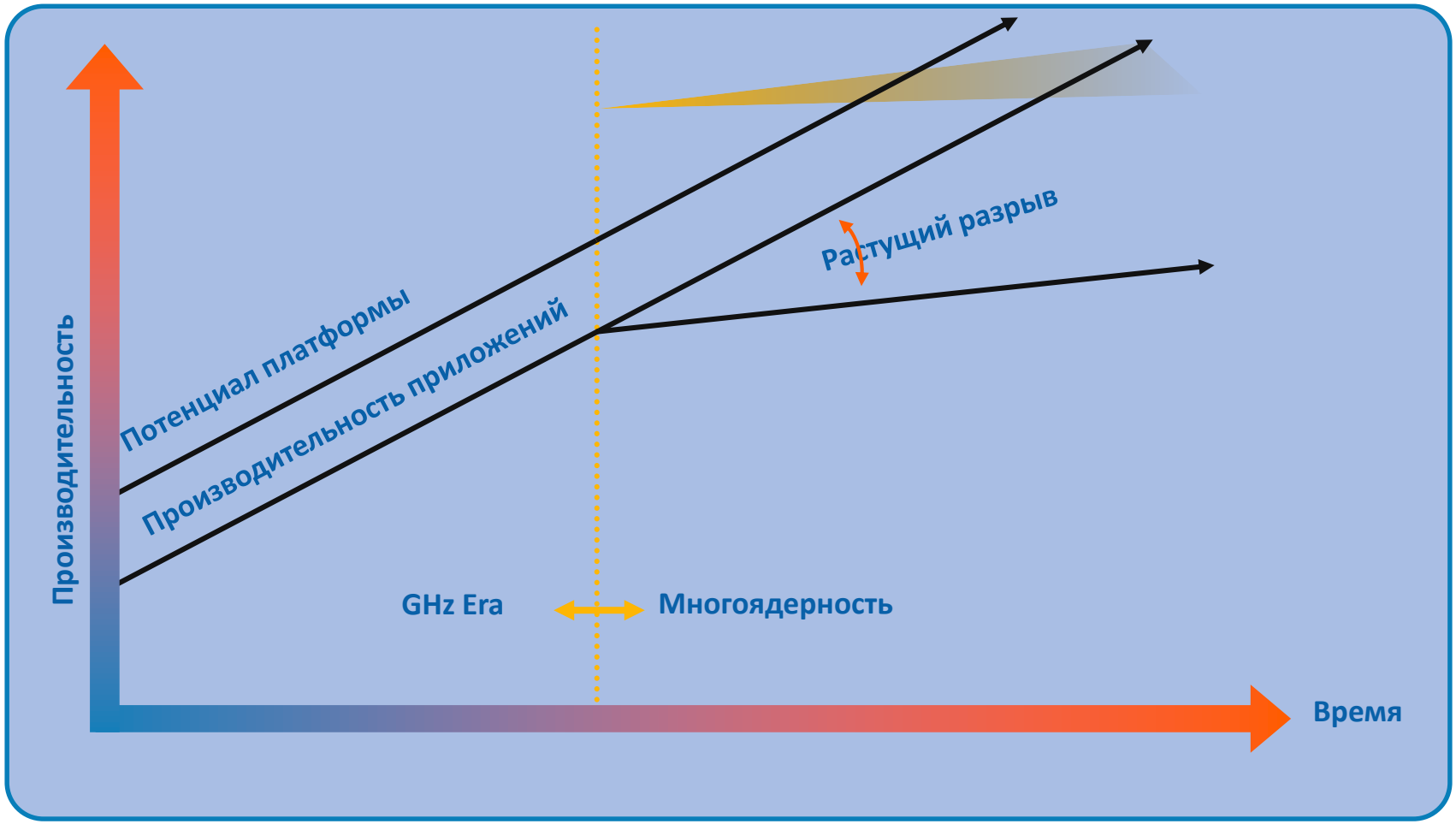
- Количество синхронизирующих тактов, поступающих извне на вход схемы за одну секунду.



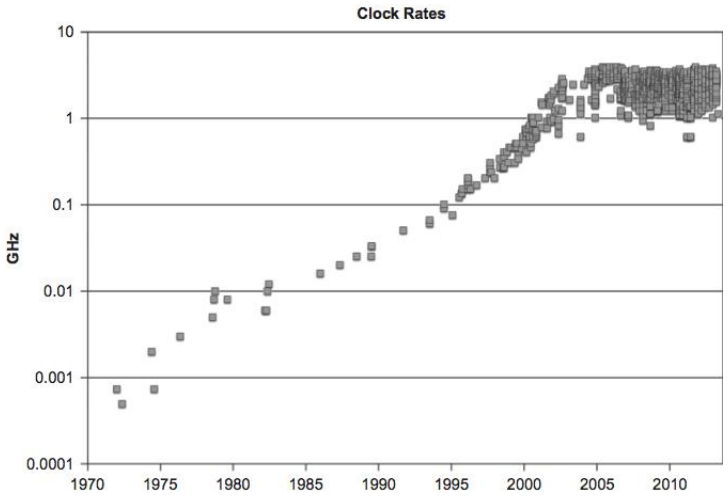
Три важных темы при изучении архитектуры

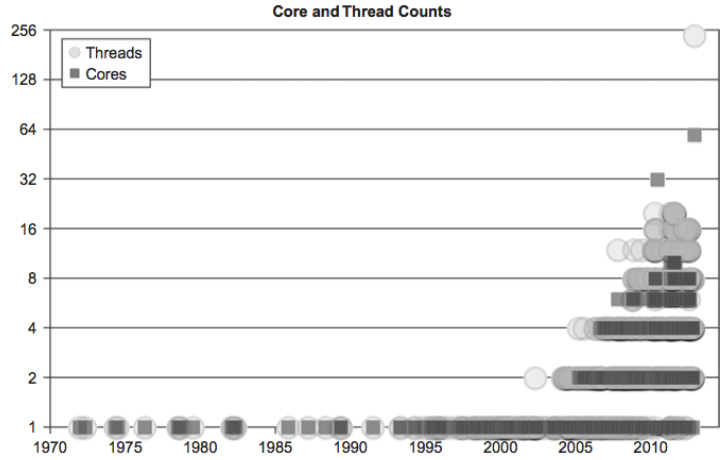
- Многоядерность
- Микроархитектура
- Система команд

Многоядерные процессоры нуждаются в параллельных приложениях



Параллелизм – норма жизни





- Мультиядерные и многоядерные процессоры
- Степень аппаратного параллелизма продолжает расти
- Требуется явный параллелизм на уровне софта

Виды параллелизма
Аналогия из спорта

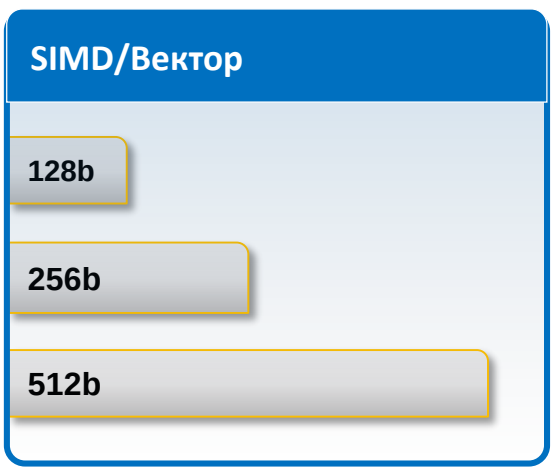
Photo credit André Zehetbauer (CC) BY-SA 2.0

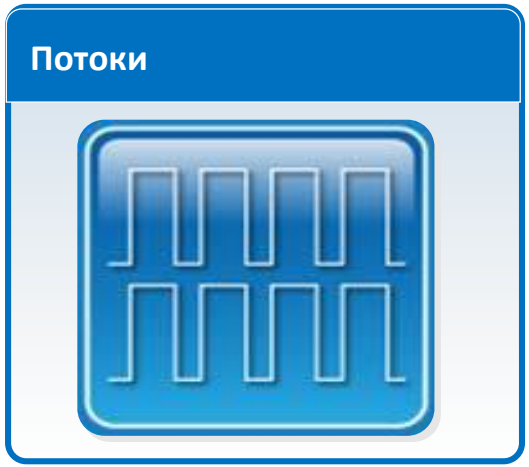
Photo credit JJ Harrison (CC) BY-SA 3.0

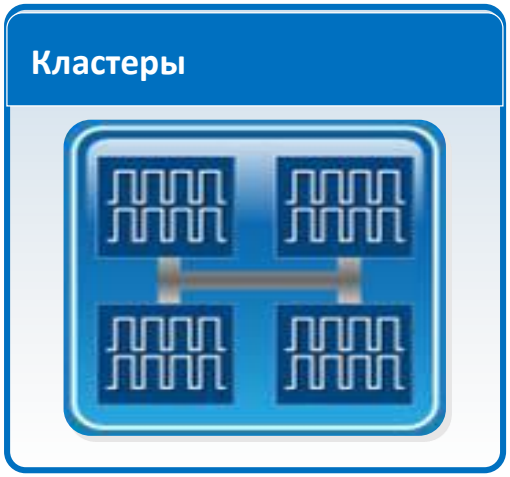
Параллелизм (независимый)

Согласованный параллелизм (взаимодействующий)

Виды параллелизма



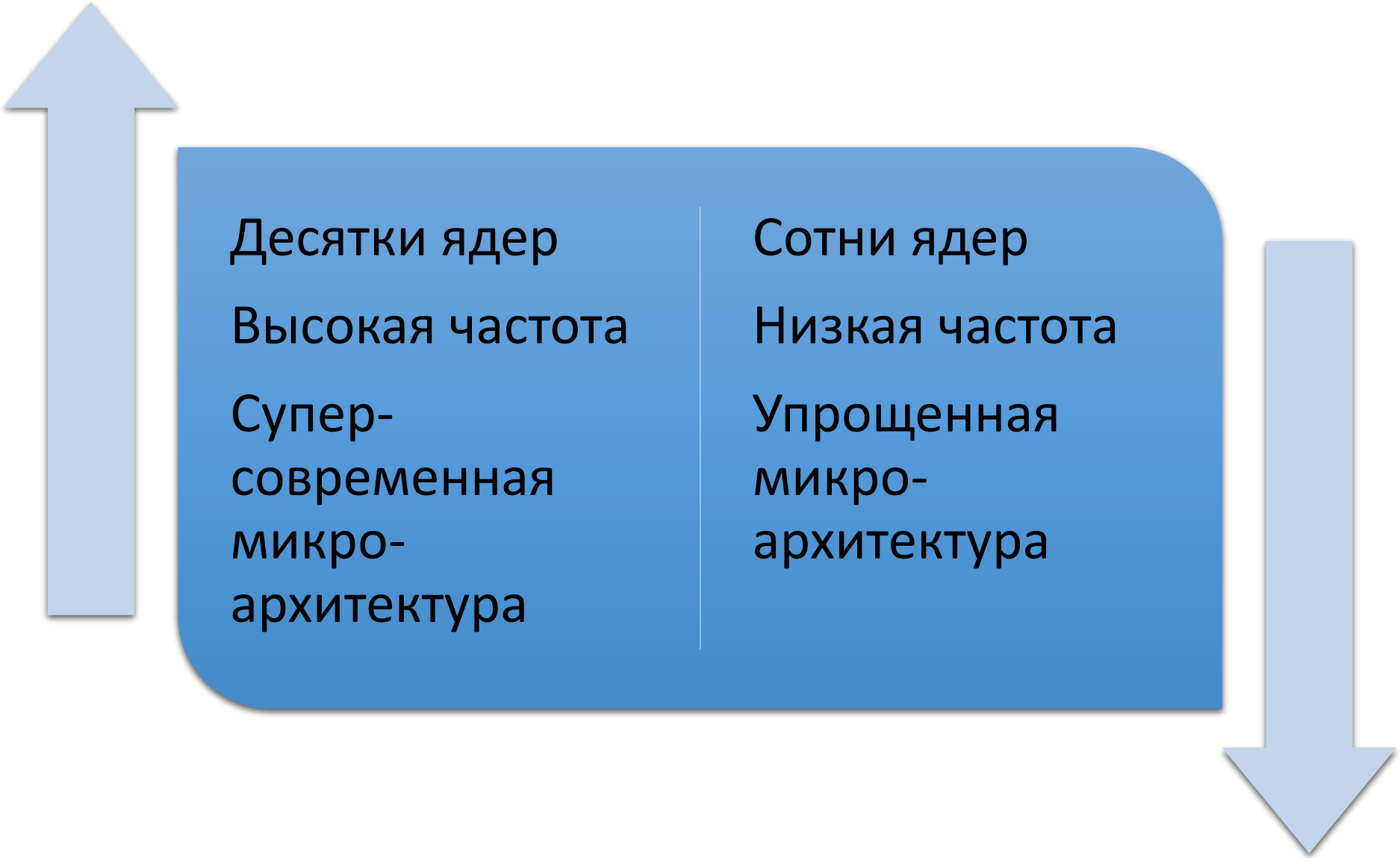




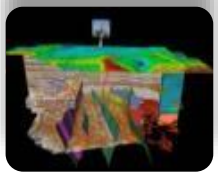
Закон Амдала

- «В случае, когда задача разделяется на несколько частей, суммарное время её выполнения на параллельной системе не может быть меньше времени выполнения самого длинного фрагмента»
- Закон Амдала показывает, что прирост эффективности вычислений зависит от алгоритма задачи и ограничен сверху для любой задачи
- Не для всякой задачи имеет смысл наращивание числа процессоров в вычислительной системе

CPU vs. GPU


Десятки ядер
Высокая частота
Супер-современная микро-архитектура
Сотни ядер
Низкая частота
Упрощенная микро-архитектура

«Параллелизм для CPU»

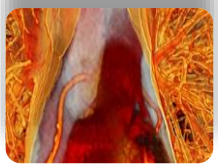


Energy & oil exploration

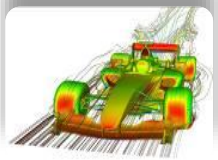
Digital content creation

Climate modeling & weather prediction

Financial analyses, trading



Medical imaging and biophysics

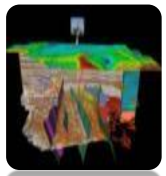


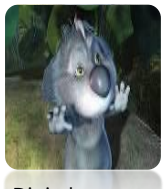
Computer Aided Design & Manufacturing

- Сложные вычислительные проблемы, которые могут быть разбиты на части, выполняющиеся параллельно
- Параллельные приложения отличаются по гранулярности и программным моделям
- Массивно параллельные приложения встречаются везде! - workstation, HPC, Data Centers
- Примеры массивно параллельных вычислений: Vector Math, FFTs, Sparse and Dense Matrix Multiplication, Convolution, LU Factorization, Sort, Monte Carlo, Black-Scholes, итд.

«Параллелизм для GPU»

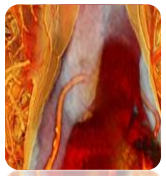
- Параллелизм по данным
- Число арифметических операций велико по сравнению с операцией над памятью
- Обработка изображений, кодирование и декодирование видео, распознавание образов

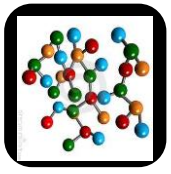

Energy – Seismic Applications

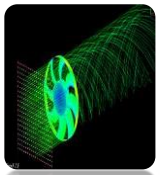

Digital content creation


Climate modeling Weather prediction

Financial Analysis Trading


Medical imaging and biophysics


Molecular Modeling


Computational Fluid Dynamics


DNA Sequencing

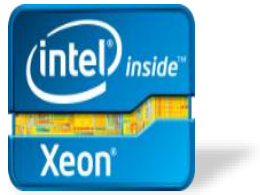

EDA

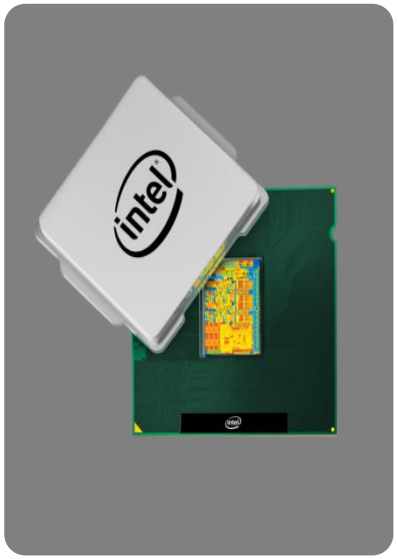

Government Defense


CAD / CAM

Дополняющие технологии

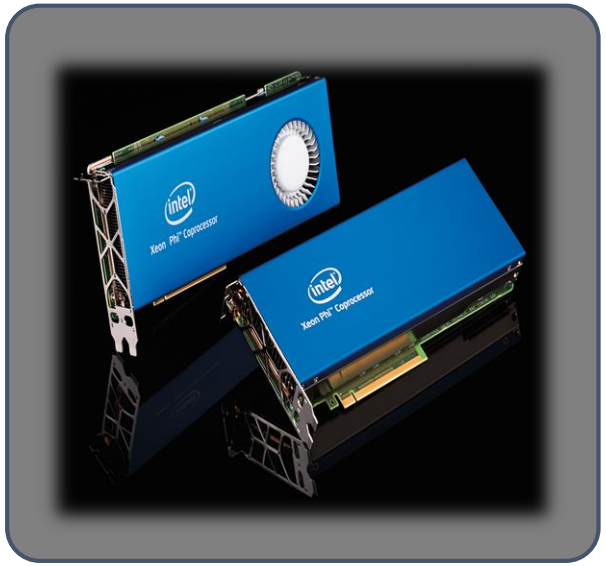
Процессор Intel® Xeon®





Общие HPC расчеты

Сопроцессор Intel® Xeon Phi™



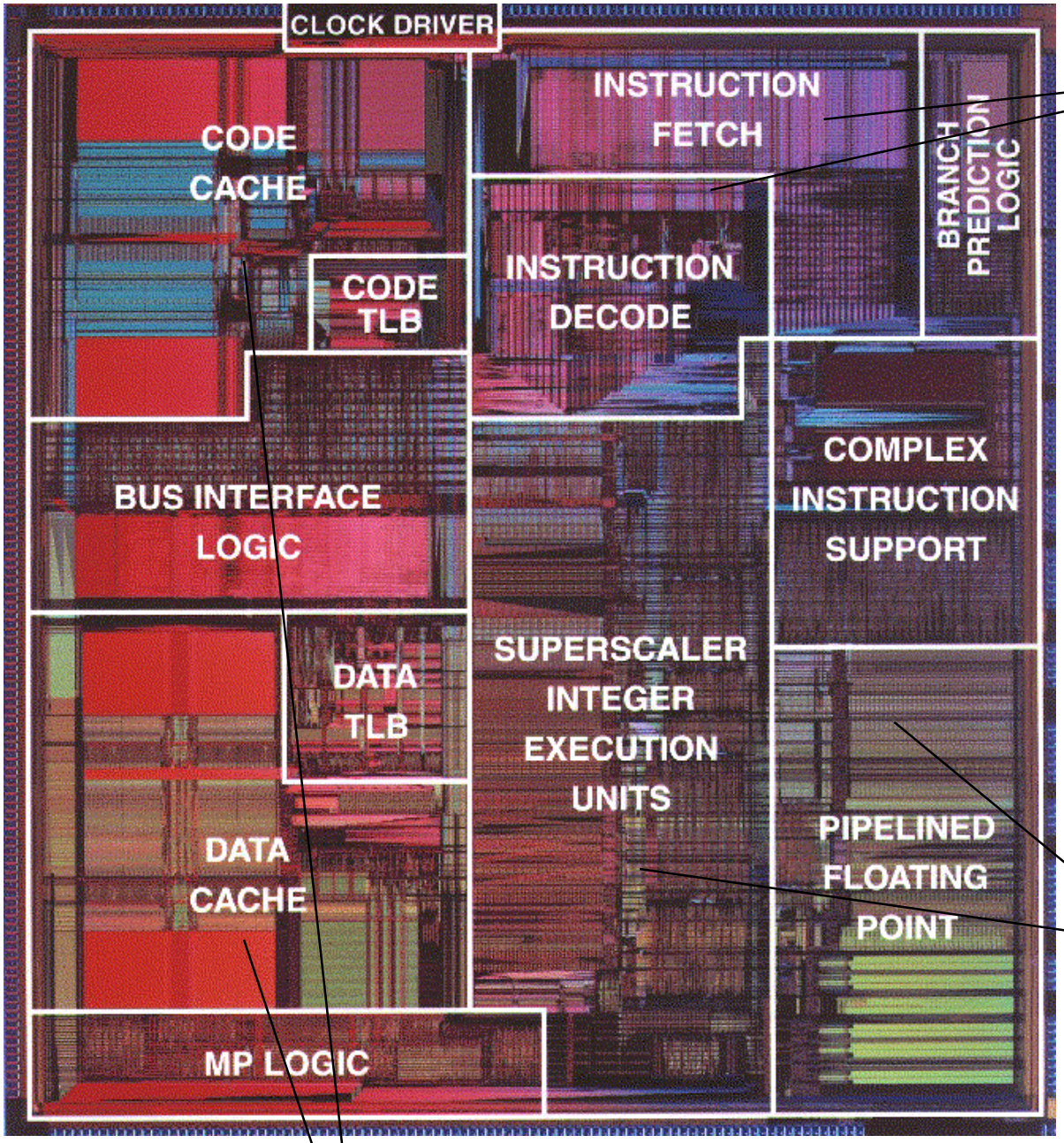
Сильно-параллельные HPC расчеты

Микроархитектура

Способ, которым данная архитектура набора команд реализована в процессоре

Общие архитектурные характеристики

Процессор Intel® Xeon® Processor E5-2690		Сопроцессор Intel® Xeon Phi™ 5110P
2.9GHz	**Частота**	1.053GHz
8 (Multi-Core)	**Ядра**	60 (Many-Core)
16	**Потоки**	240
256	**SIMD**	512
Когерентный	**Кэш**	Когерентный
Общая память	**Память**	Общая память



Control Unit

A Typical Microprocessor Layout: The Intel Pentium Classic

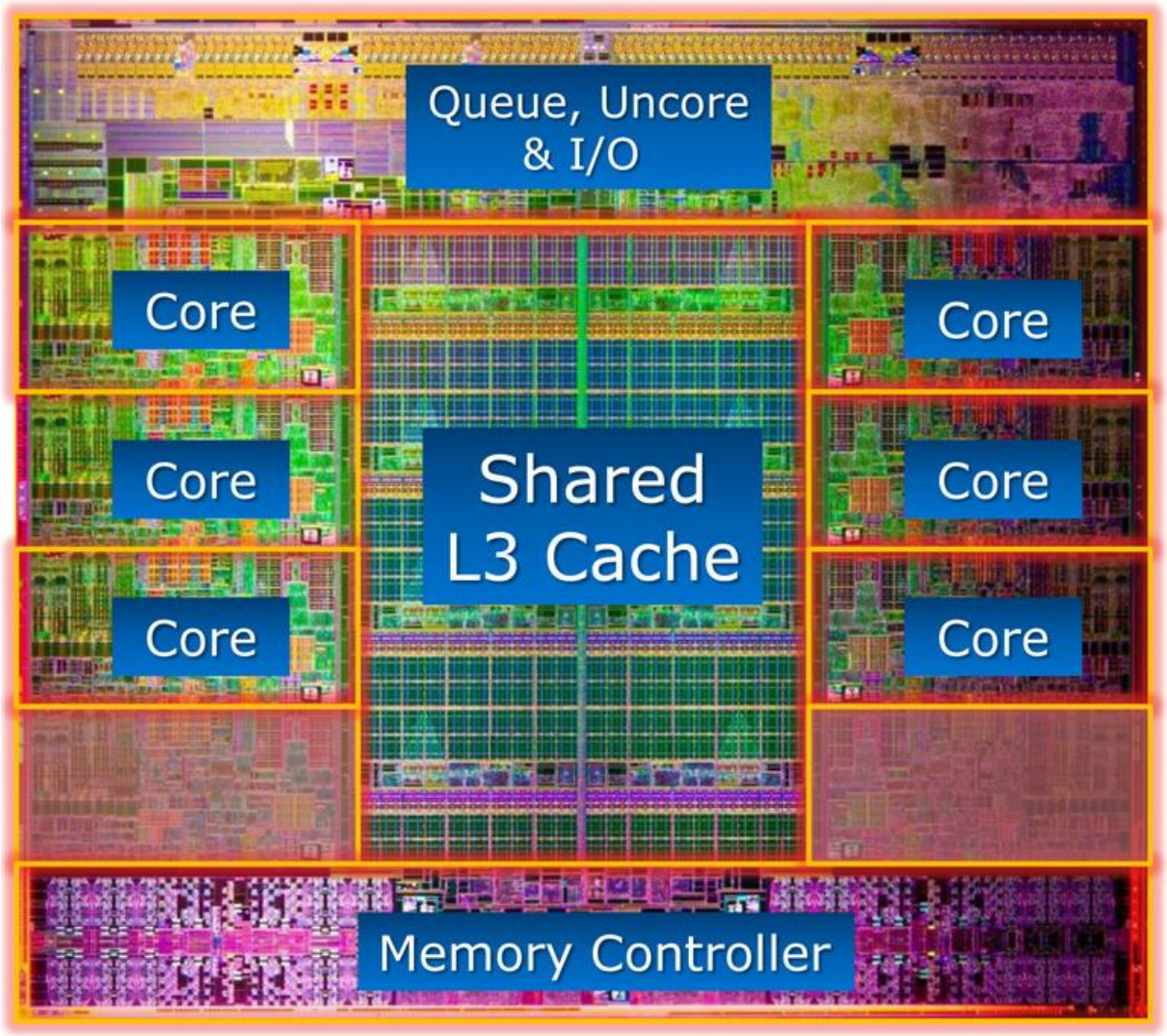
1993 - 1997

60MHz - 233 MHz

Datapath

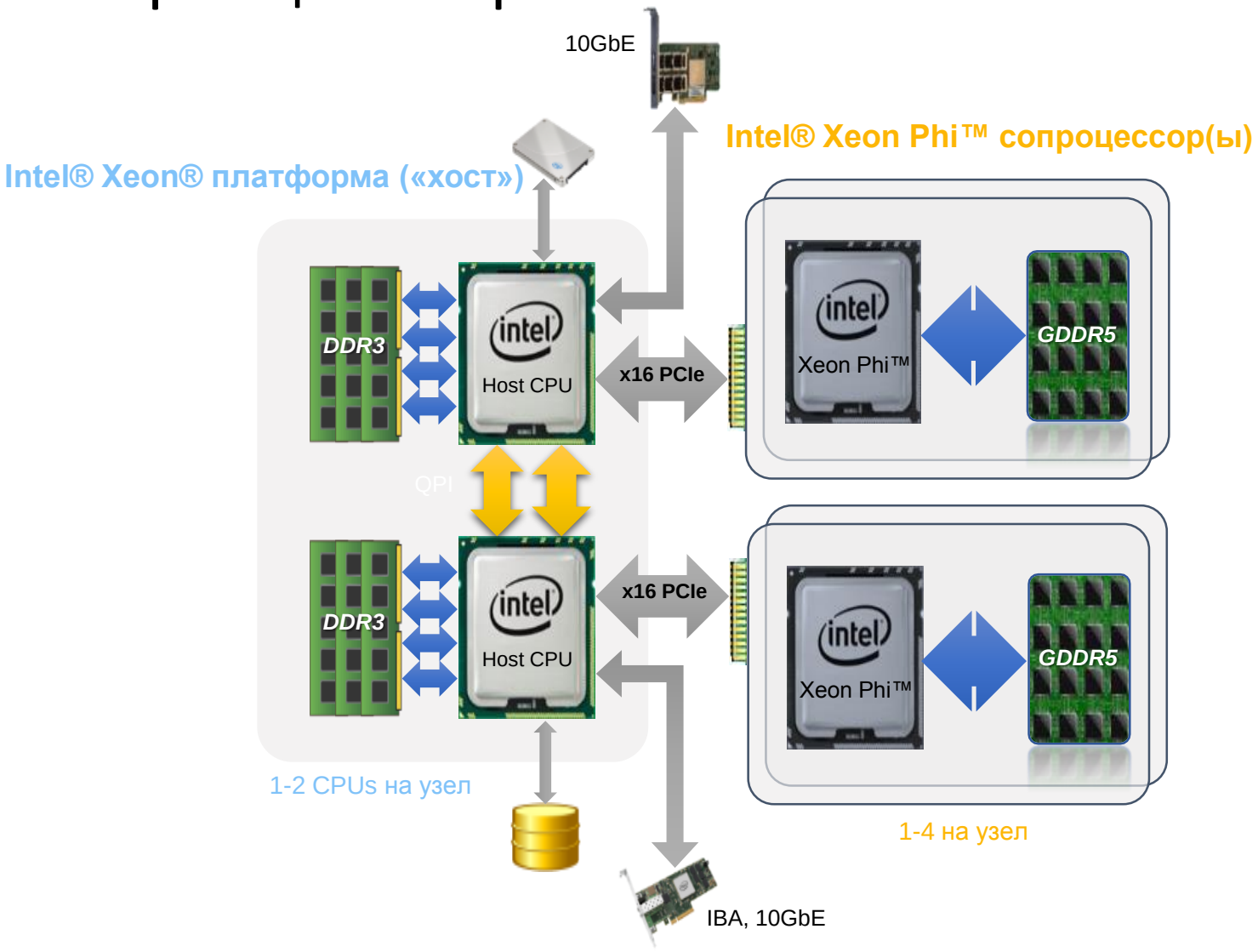
First Level of Memory (Cache)

Многоядерный микропроцессор

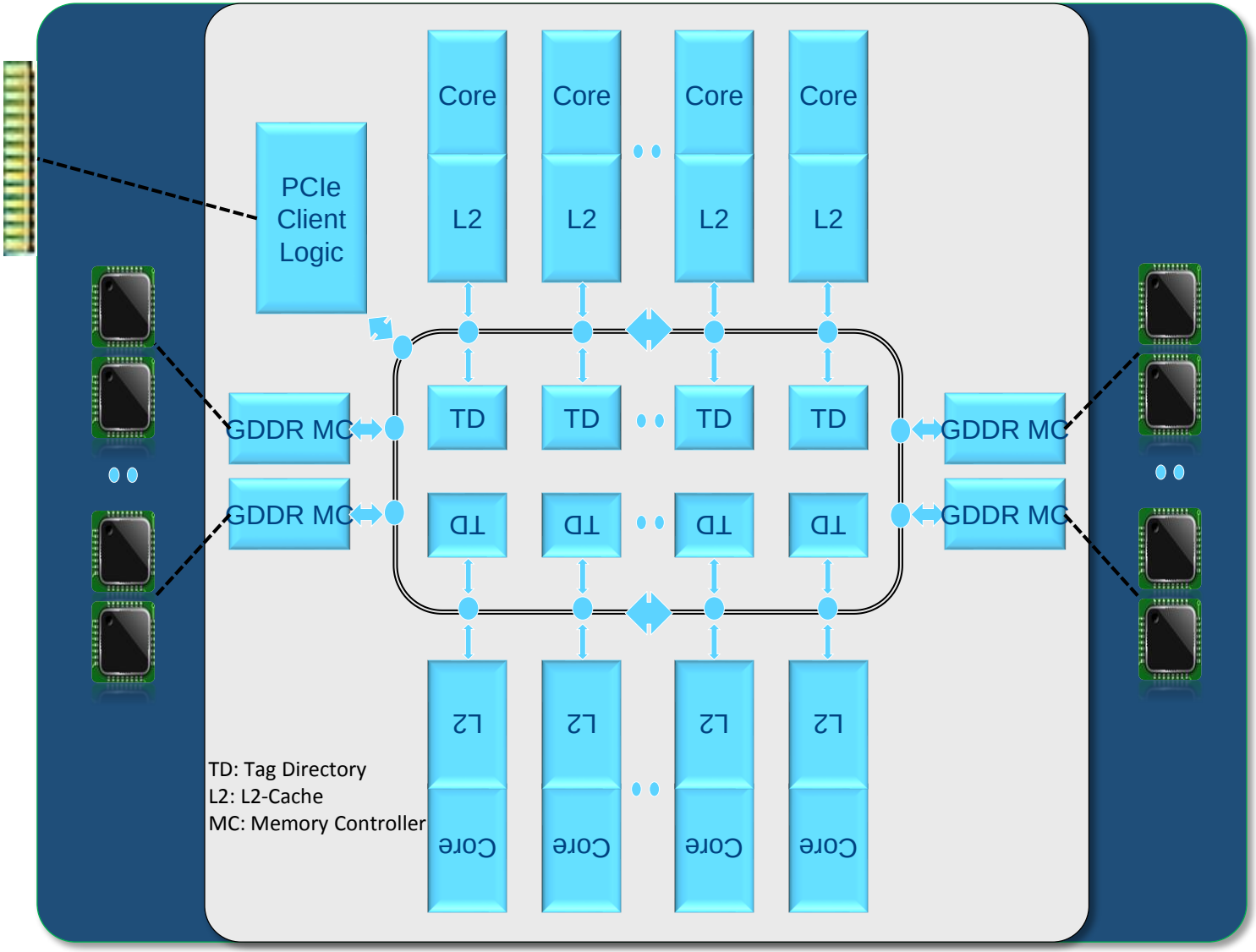


Intel Core i7-3960X

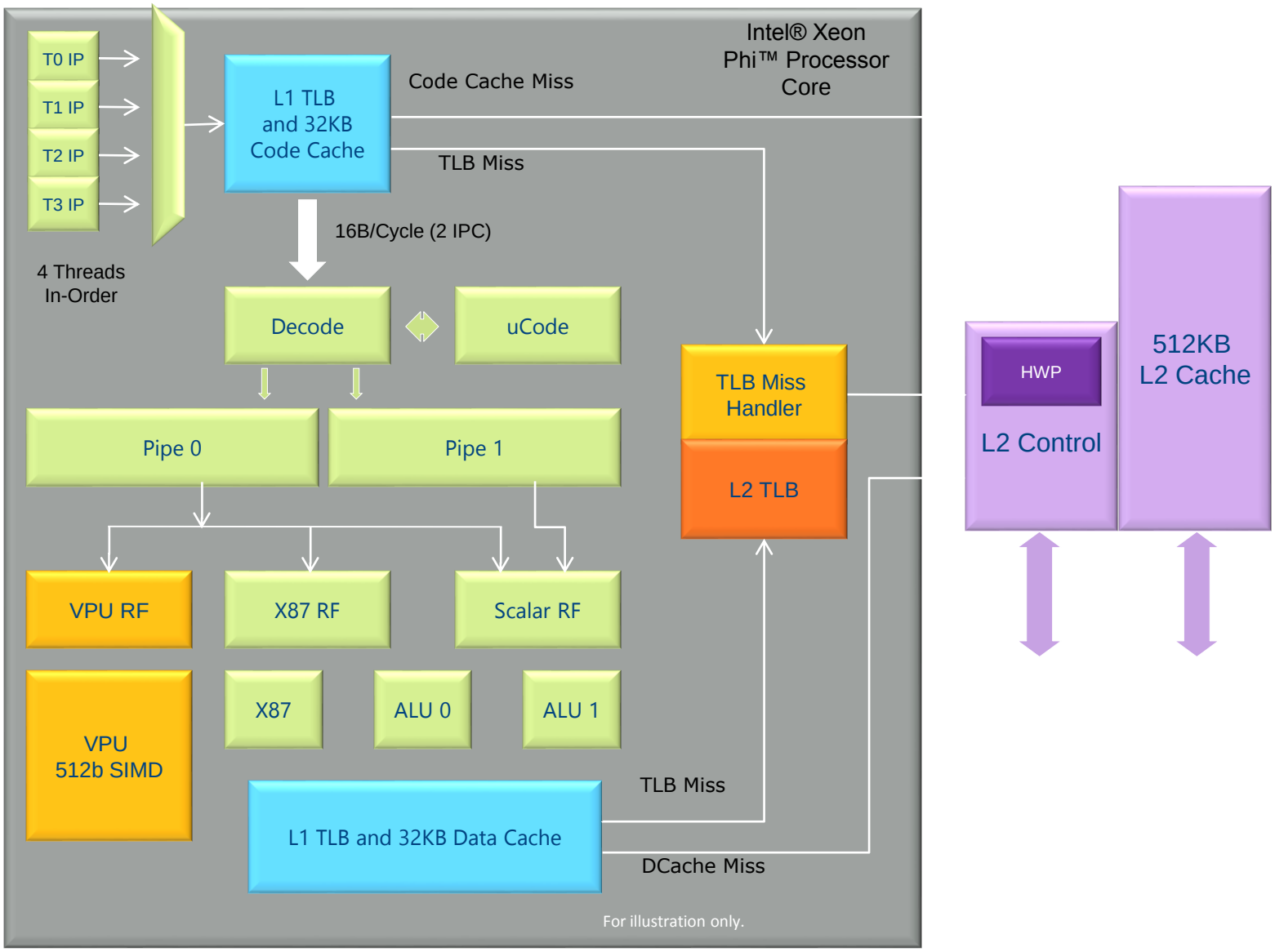
Типичная платформа с сопроцессором Intel® Xeon Phi



Обзор микроархитектуры Intel® Xeon Phi™



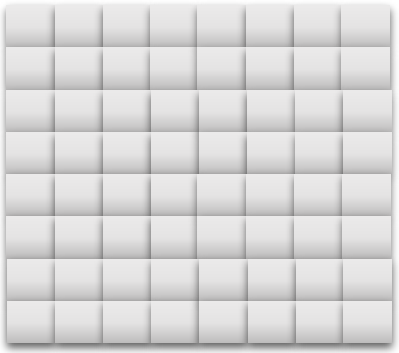
Ядро Intel® Xeon Phi™



Эффективные приложения для Intel® Xeon Phi™

- Допускают массовый параллелизм
- Имеют высокую вычислительную сложность
  - Векторизация
  - Большое количество вычислений на единицу данных
- Умещаются в доступную память

Multicore
(8+)



Many-Core
(60)

Система команд

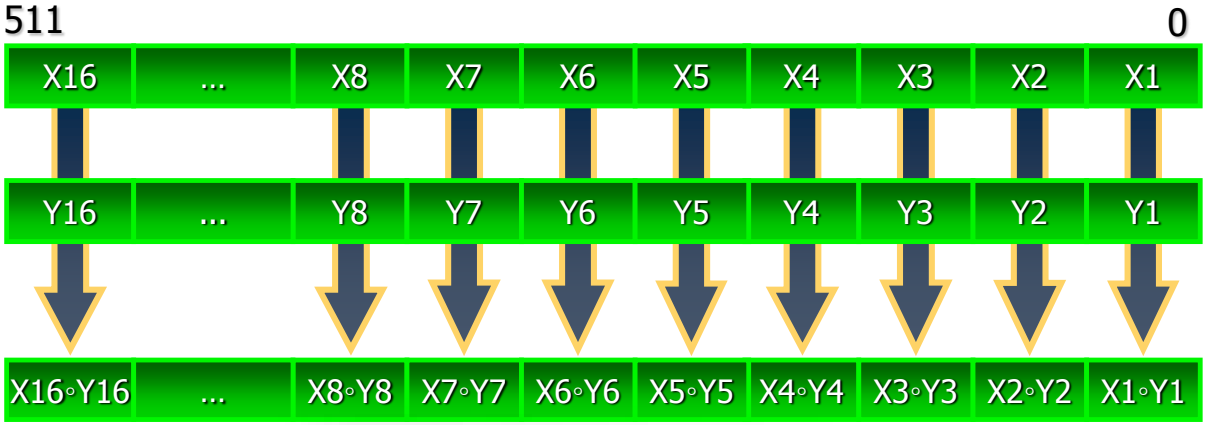
Соглашение о предоставляемых архитектурой средствах программирования, а именно: определённых типах данных, инструкций, системы регистров, методов адресации, моделей памяти, способов обработки прерываний и исключений, методов ввода и вывода

More cores. Wider vectors. Co-Processors.

Images do not reflect actual die sizes

	Intel® Xeon® processor 64-bit	Intel® Xeon® processor 5100 series	Intel® Xeon® processor 5500 series	Intel® Xeon® processor 5600 series	Intel® Xeon® processor code-named Sandy Bridge	Intel® Xeon® processor code-named Ivy Bridge	Intel® Xeon® processor code-named Haswell	Intel® Xeon Phi co-processor Knights Corner
Core(s)	1	2	4	6	8			60
Threads	2	2	8	12	16			240
SIMD Width	128	128	128	128	256	256	256	512
	SSE2	SSSE3	SSE4.2	SSE4.2	AVX	AVX	AVX2 FMA3 TSX	

SIMD/Параллелизм по данным

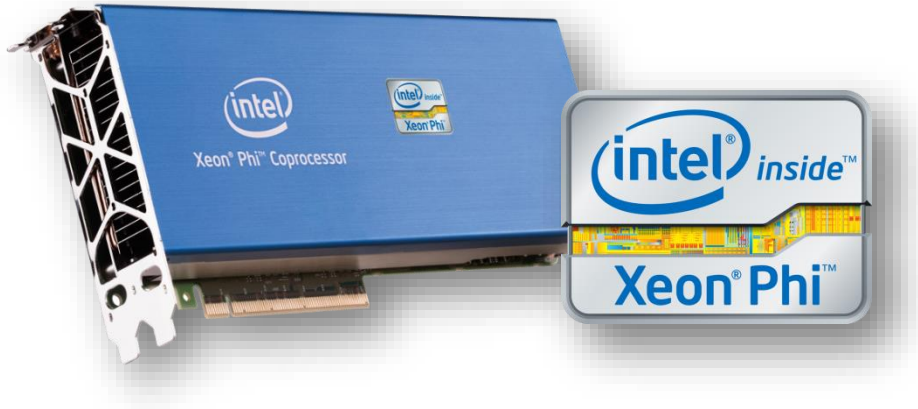


Intel Xeon Phi

Вектор: **512 bit**

Типы:

- integer (32 и 64 бит)
- float (F32)
- double (F64)



SIMD, Single Instruction Multiple-Data

Векторизация кода

- Заставляет последовательный код использовать возможности параллелизма по данным (SIMD) процессоров Intel
  - Вручную за счет спец синтаксиса
  - Автоматически за счет компилятора

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

Почему важна векторизация ?

```
#define MAX(x,y) ((x)>(y)?(x):(y))
#define MIN(x,y) ((x)<(y)?(x):(y))
#define SAT2SI16(x) \
  MAX(MIN((x),32767),-32768)

void foo1(int n, short *A, short *B){
int i;
#pragma ivdep
#pragma vector aligned
  for (i=0; i<n; i++)
    A[i] = SAT2SI16(A[i]+B[i]);
}
```

Скалярный код:

```
• movsx     r11d, [rdx
• movsx     ebx, [r8+r
• add       r11d, ebx
• cmp       r11d, 32767
• cmovge    r11d, eax
• cmp       r11d, -32768
• cmovl     r11d, ecx
• mov       [rdx+r9*2], r11w
• inc       r9
• cmp       r9, r10
• jb        .B1.8
```

11 инстр./ 1 элем

Векторный код (SSSE-3):

```
movdqa    xmm0, [rdx+rax*2]
paddsw    xmm0,  [r8+rax*2]
movdqa    [rdx+rax*2], xmm0
add       rax, 8
cmp       rax, r9
jb        .B1.4
```

6 инстр/ 8 элем

2. Модели и программы

Что и на чем пишут для (супер)компьютера?

Как модели и программы связаны с архитектурой?

Параллелизм на всех уровнях

Instruction Parallelism

Data Parallelism

Thread Parallelism

Cluster / Process Parallelism

Один поток

Улучшение скалярной производительности, компиляторы C/C++,FORTRAN, высокопроизводительные библиотеки, инструменты анализа производительности

Уровень узла

OpenMP, Ct , CILK

Распределенная система/ Кластер

Cluster Tools, Cluster OpenMP, MPI Checker

Будущие процессоры должны поддерживать существующую базу приложений и минимизировать затраты на оптимизацию

Гетерогенное программирование

Инструменты

MKL

Fortran (CAF)

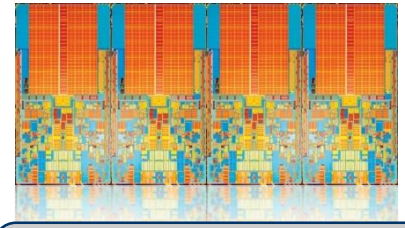
TBB

OpenMP

C++

Cilk Plus

OpenCL



Исполняемый файл для CPU

PCIe

Исполняемый файл для сопроцессора Xeon Phi

Инструменты

MKL

Fortran (CAF)

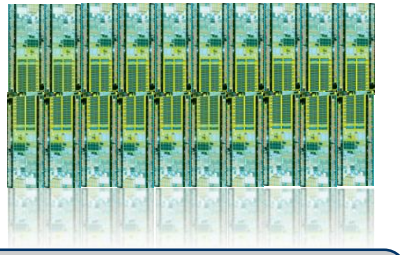
TBB

OpenMP

C++

Cilk Plus

OpenCL



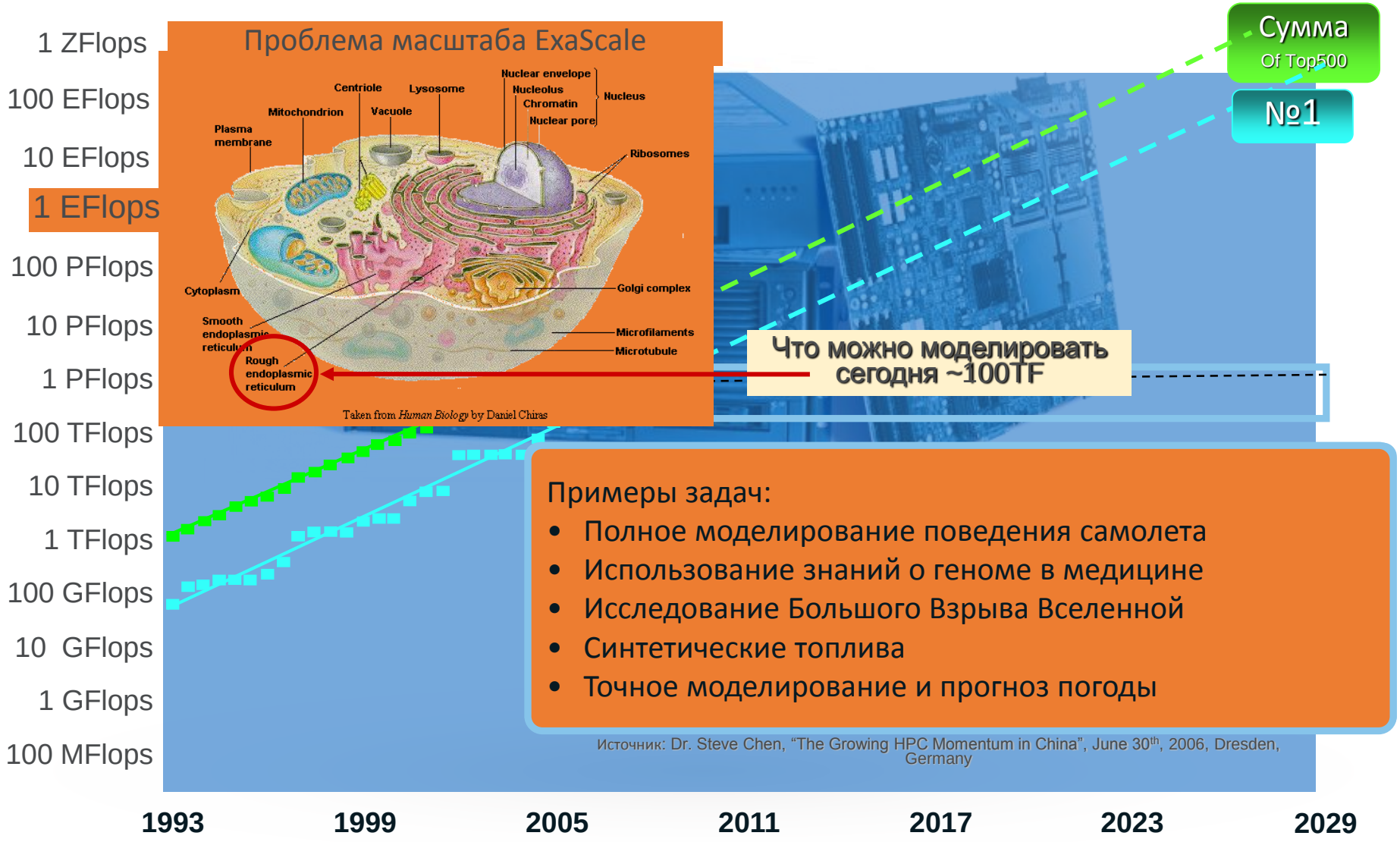
Параллельное программирование одно и то же

3. Пользователи и задачи

Кому и зачем нужен (супер)компьютер?

Как пользователи и задачи связаны с архитектурой?

Проблемы и задачи



От задач - к функциям и примитивам

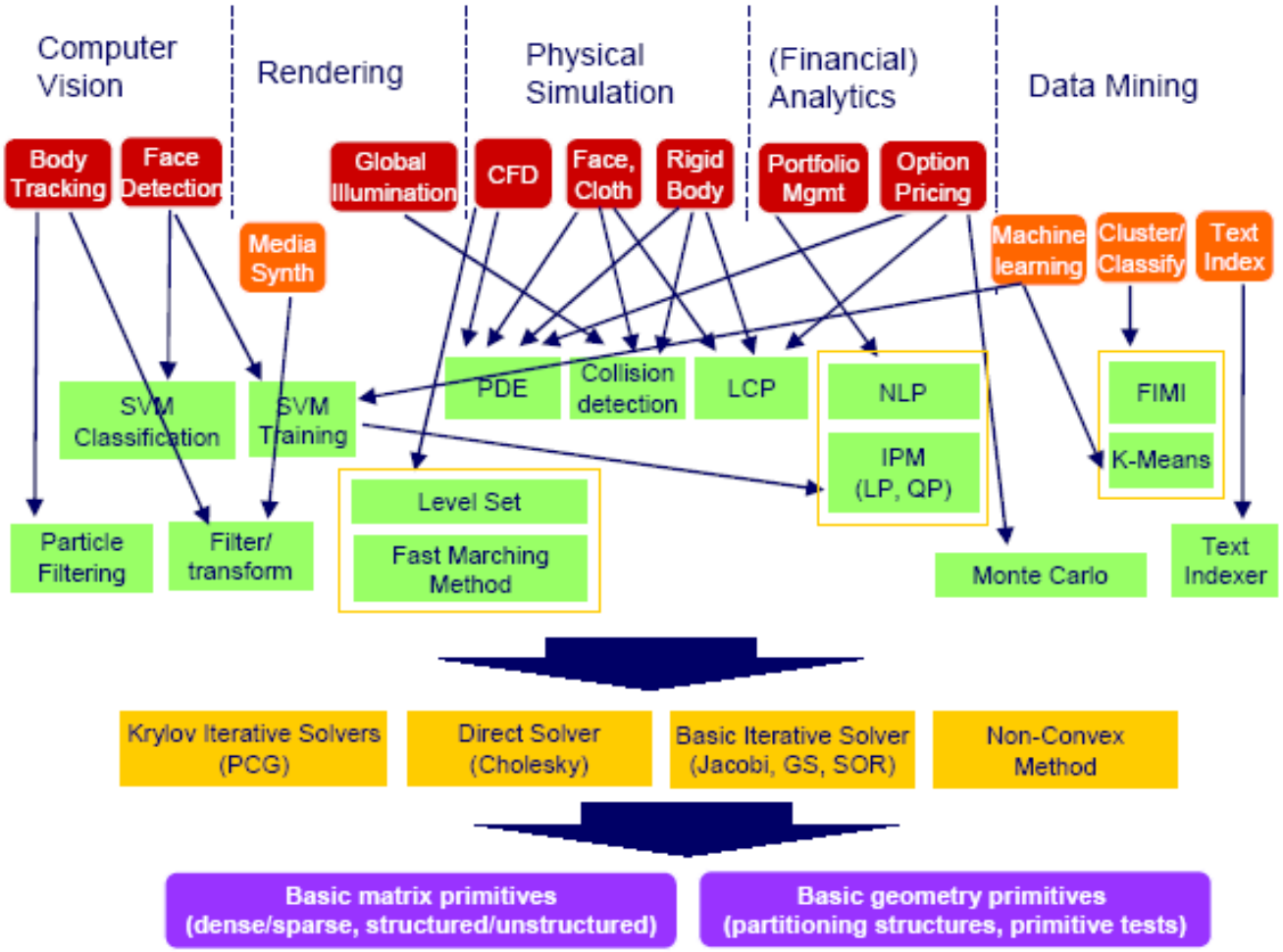


Figure 5. Intel's RMS and how it maps down to functions that are more primitive. Of the five categories at the top of the figure, Computer Vision is classified as Recognition, Data Mining is Mining, and Rendering, Physical Simulation, and Financial Analytics are Synthesis. [Chen 2006]

AMBER WRF VISIT VASP UTBENCH SU2 SG++ SeisSol ROTOR SIM R Quantum Espresso Optimized integral OPENMP/ MPI Openflow NWChem

AVBP (Large Eddy)

NEMO5

MPAS

Blast

Mardyn

BUDE

MACPO

CAM-5

Ls1

CASTEP

Harmonie

CESM

GTC

CFSv2

GS2

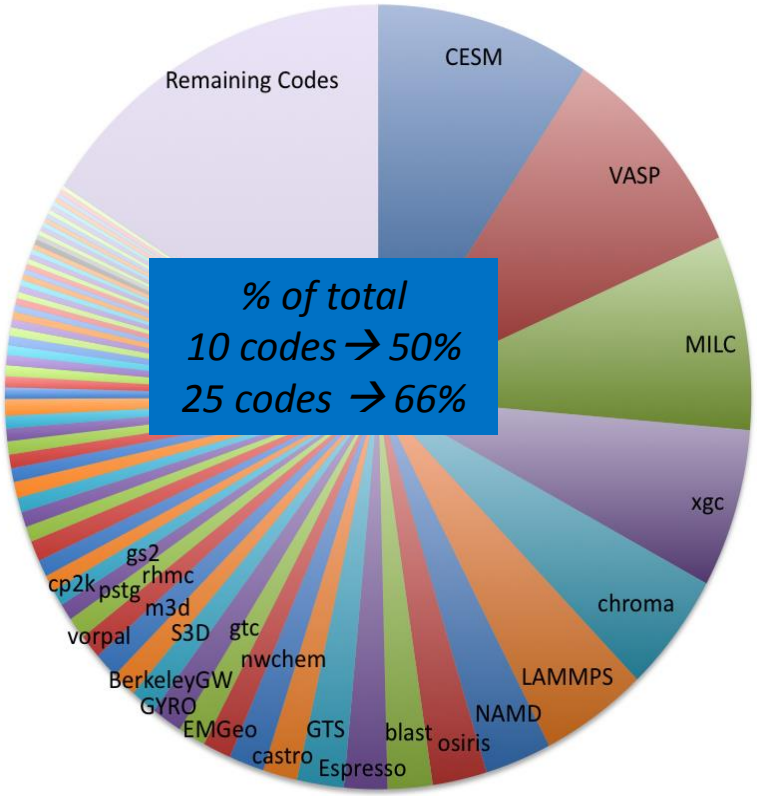
CIRCAC

Gromacs

GPAW

CliPhi (COSMOS) COSA Cosmos codes DL-MESO DL-Poly ECHAM6 Elmer FrontFlow/Blue Code GADGET GAMESS-US

Breakdown of Application Hours
NERSC - Hopper 2012[1]



Intel® Parallel Computing Centers

Collaborating to accelerate the pace of discovery

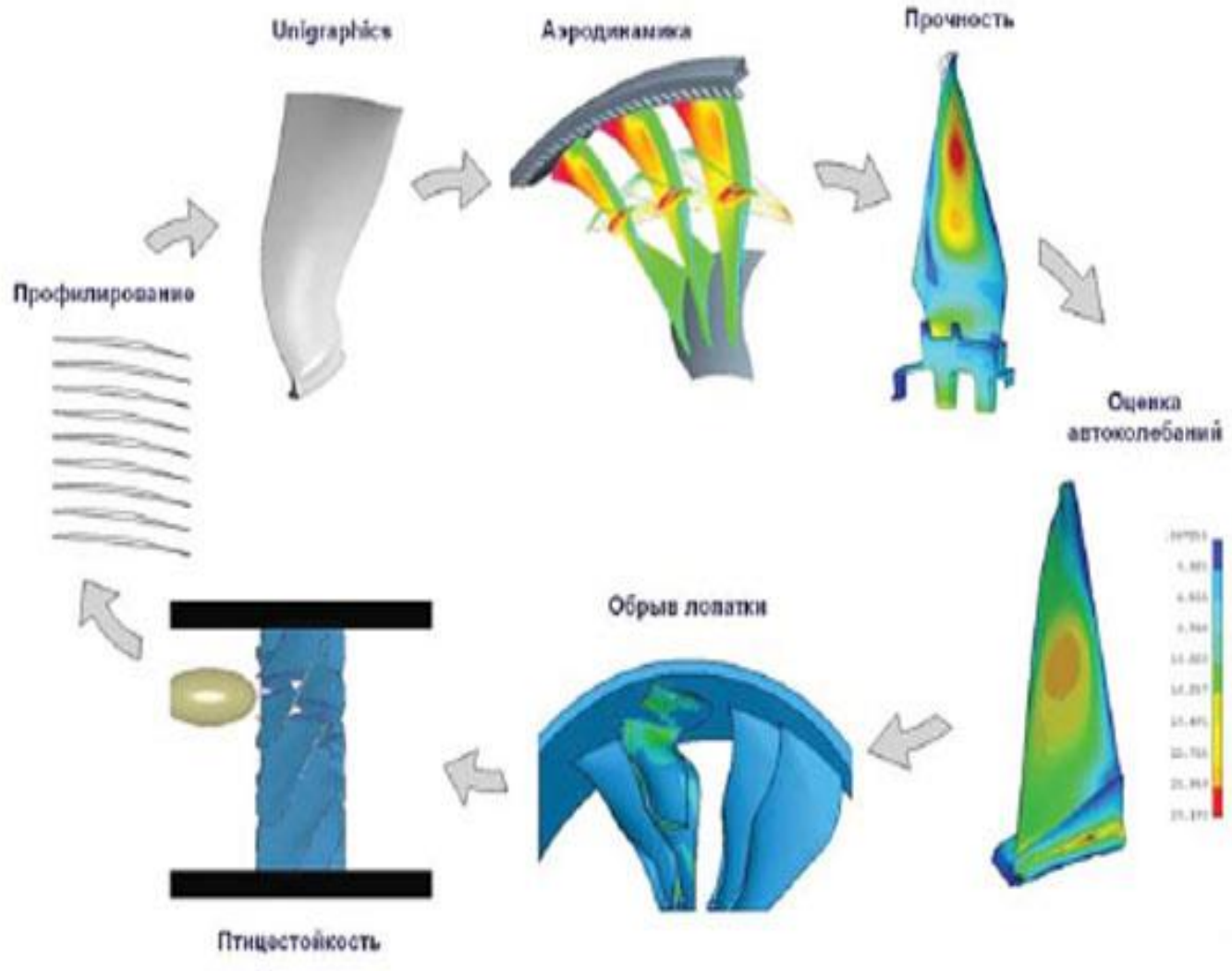
>40 Centers
13 Countries
>70 Codes
2 User Groups

https://software.intel.com/en-us/ipcc

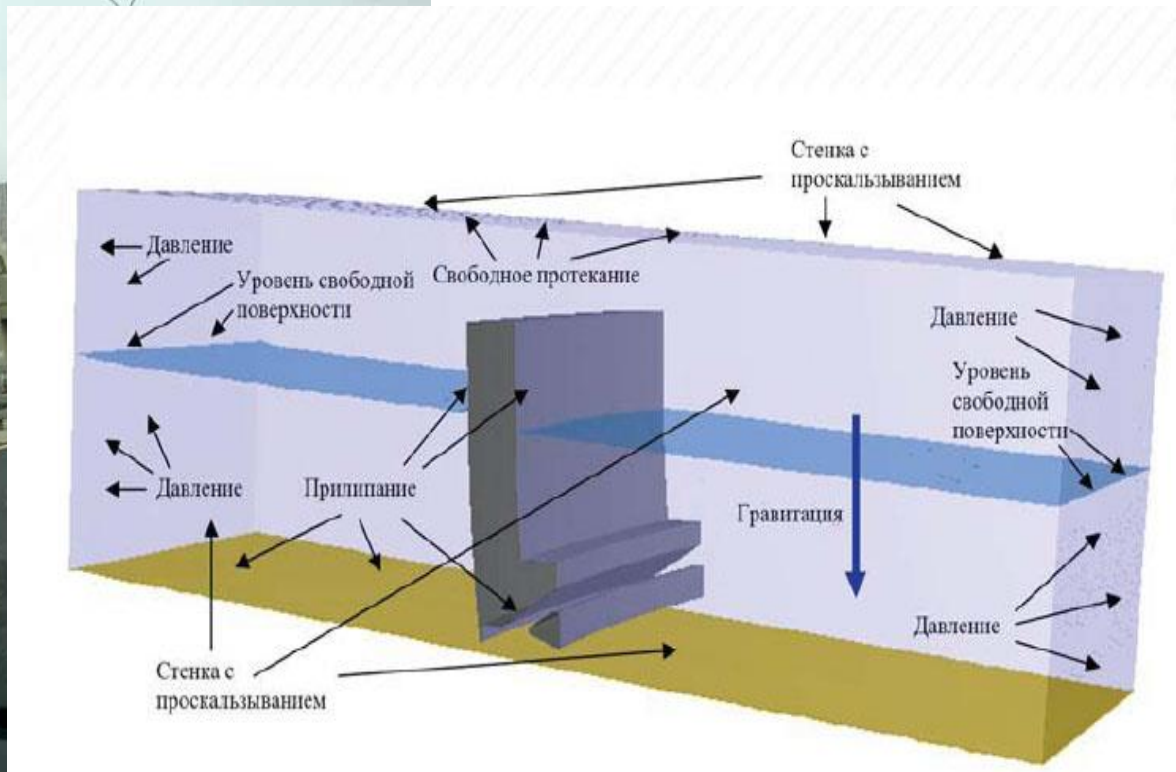
[1] *Source: NERSC*

[2] www.ihpcc2014.com

Моделирование в газодинамике

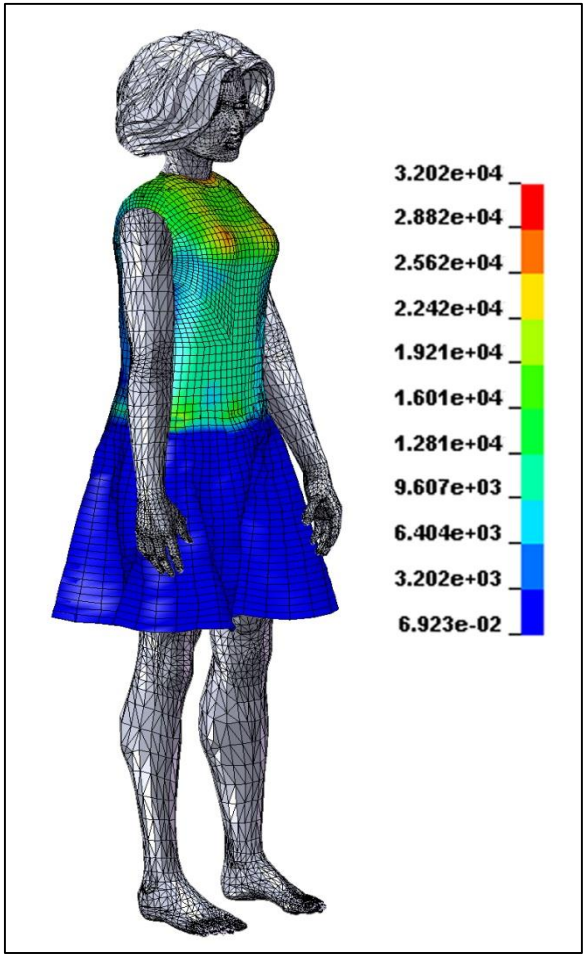


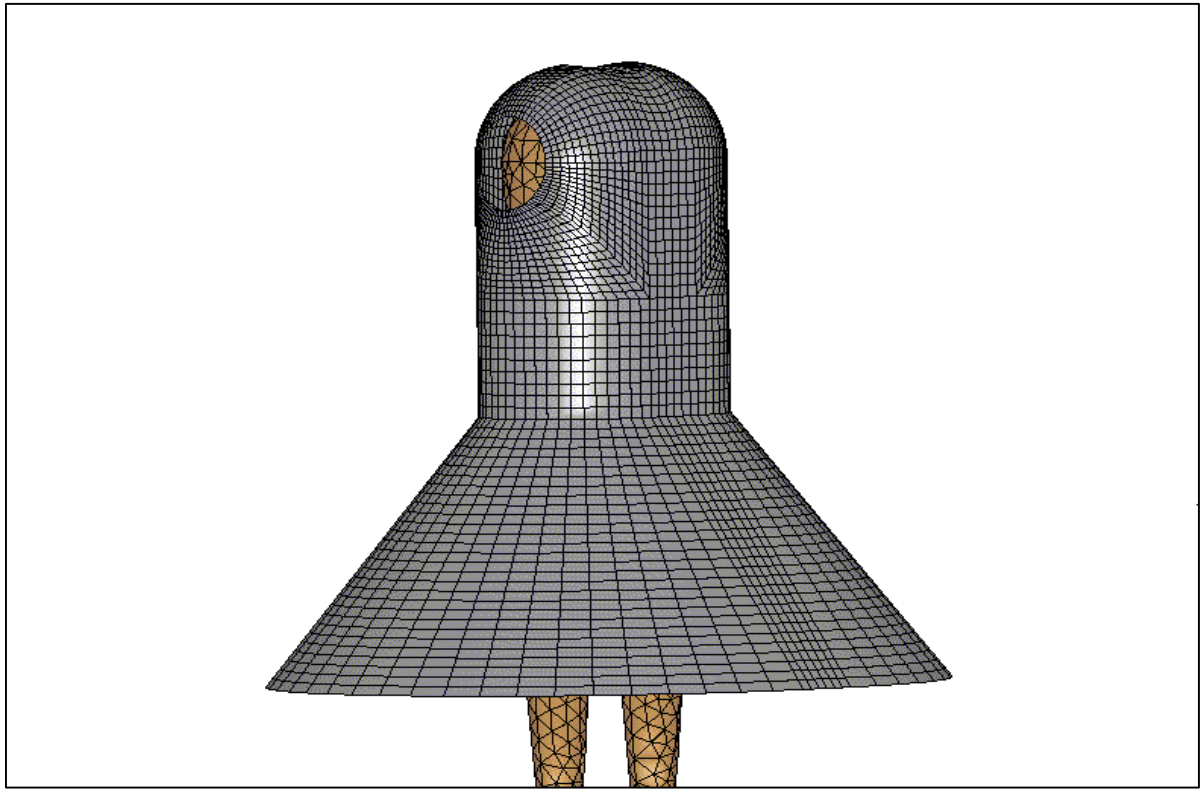
Моделирование в гидродинамике



Трехмерная модель батопорта

Моделирование в легкой промышленности



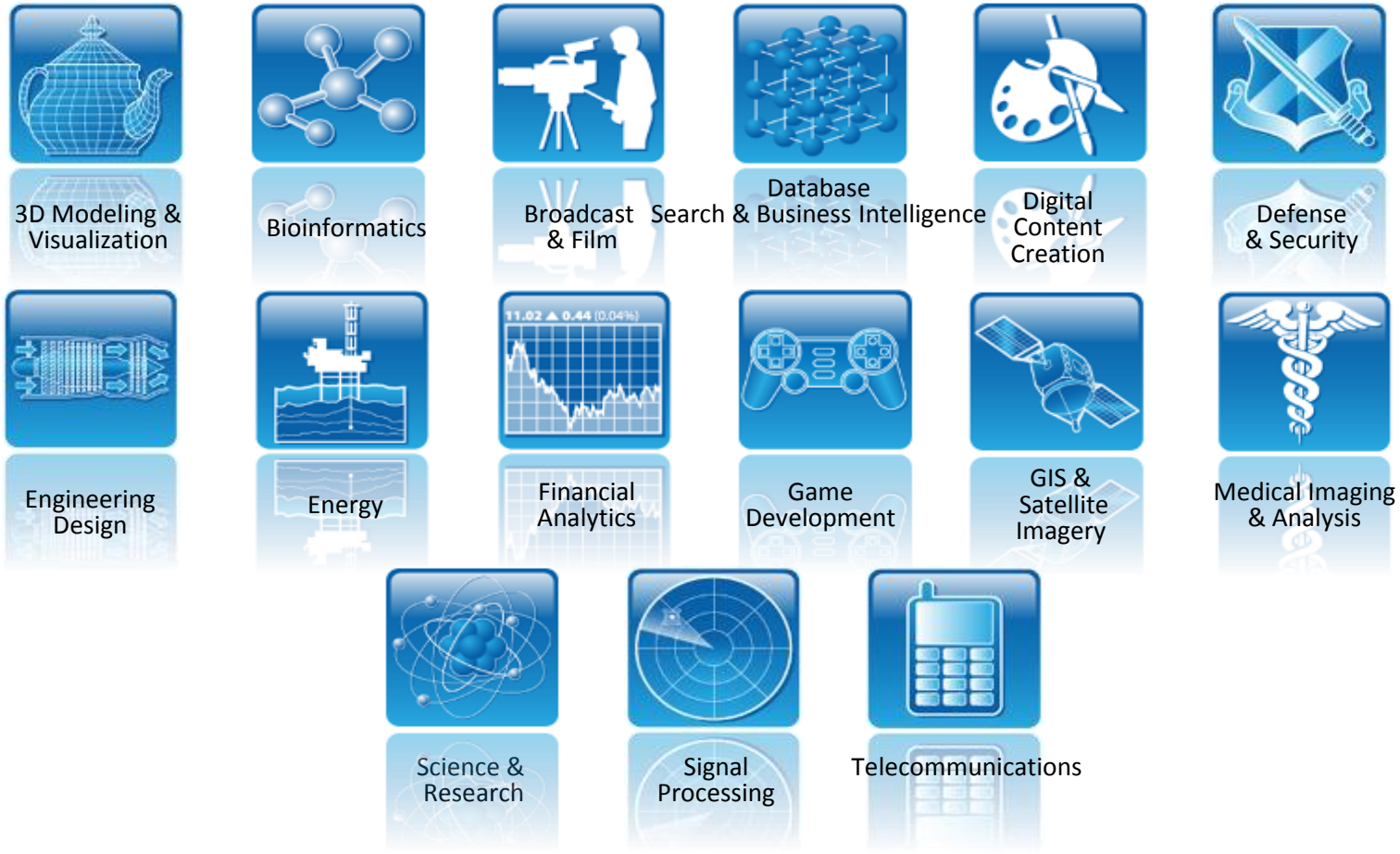


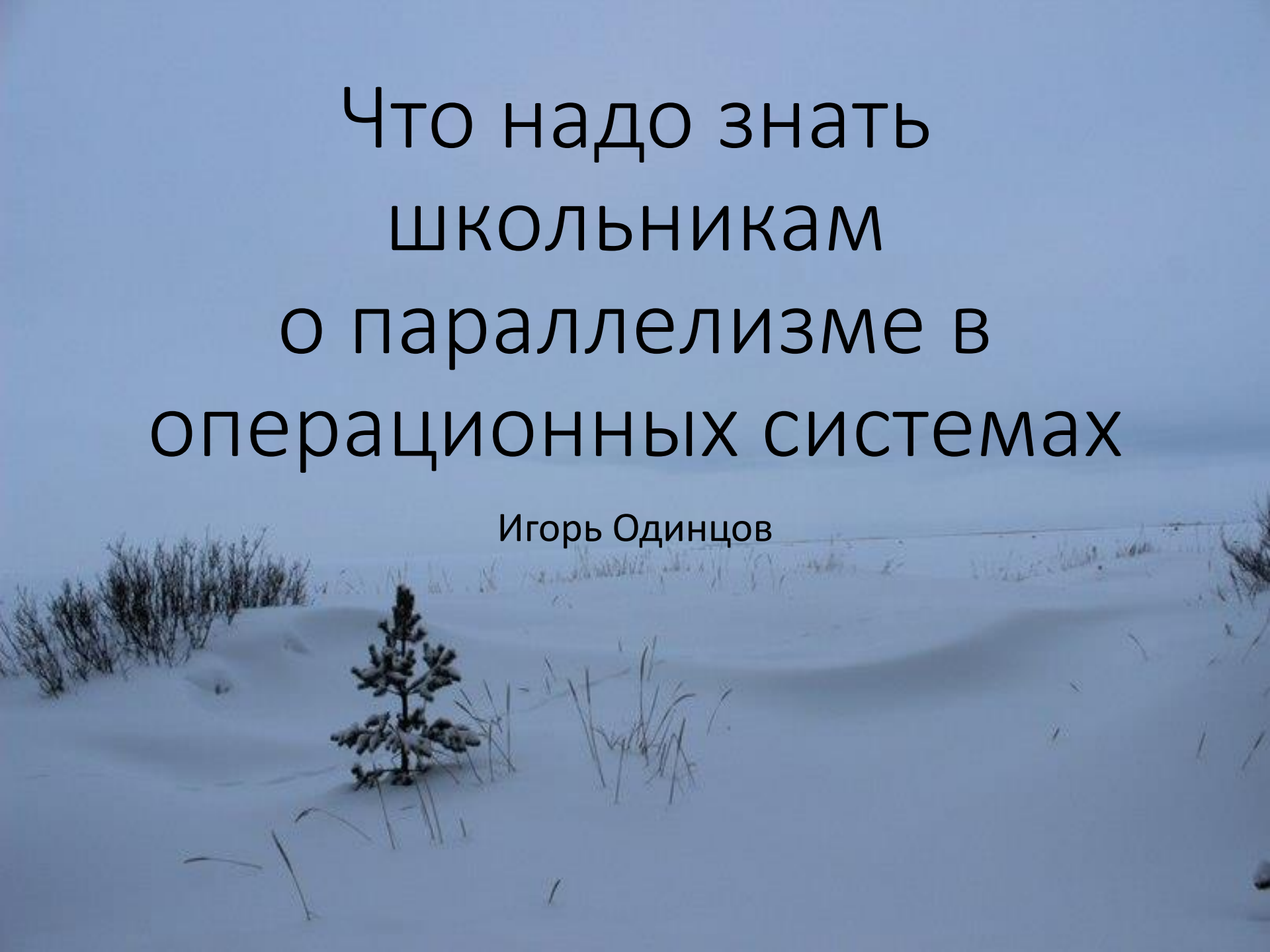
A Very Good Kitty, Indeed

DreamWorks Animation's *Puss in Boots* Uses Intel® Math Kernel Library to Help Create Dazzling Special Effects

BY GARRET ROMAINE

Итог: кто получает преимущество от суперкомпьютеров?

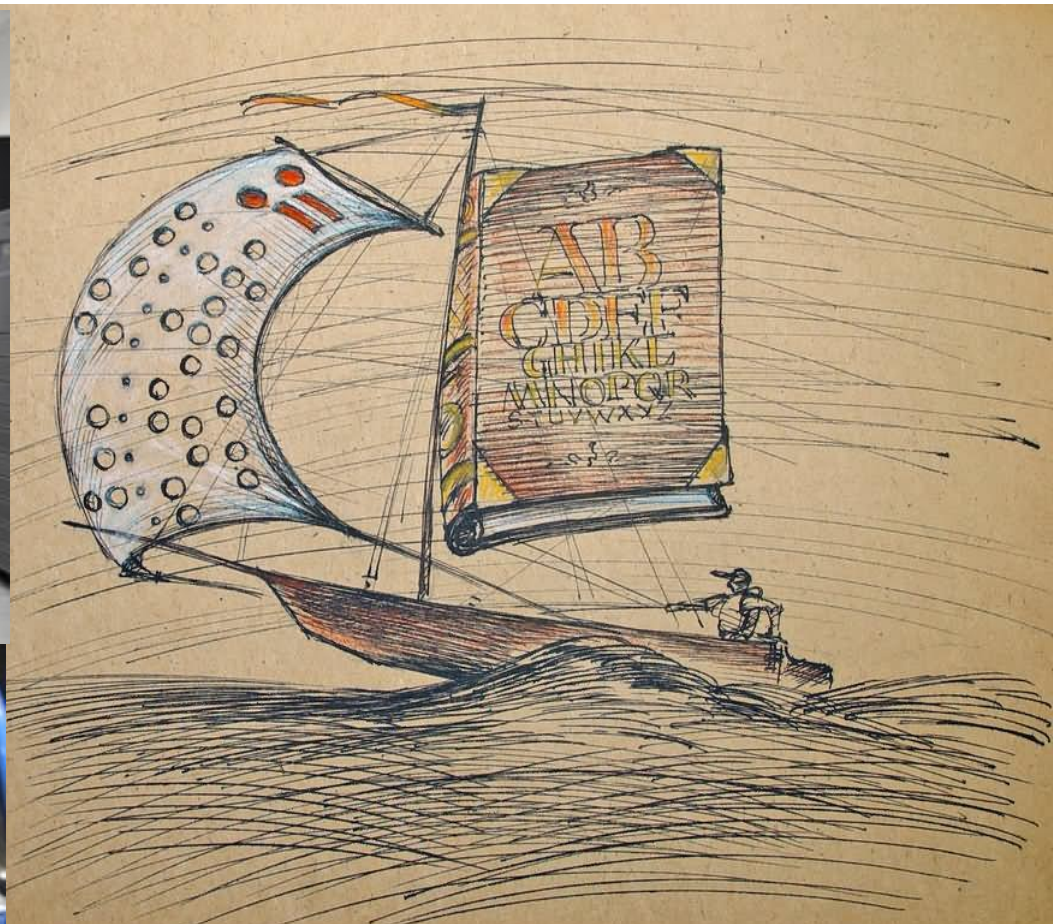




Что надо знать школьникам о параллелизме в операционных системах

Игорь Одинцов

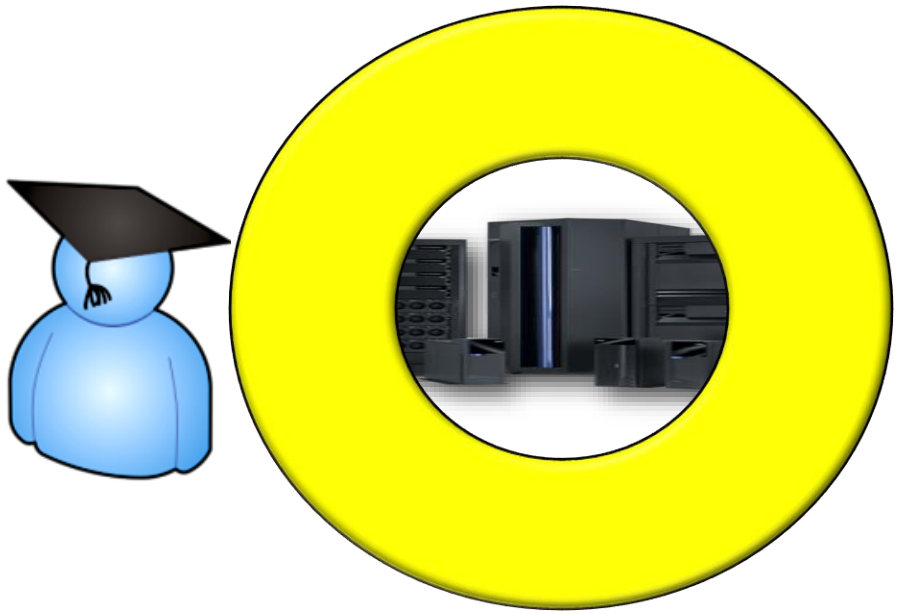
«Вычислительная машина ... приведет к формированию нового интеллектуального фона, новой операционной обстановки, органически и естественно используемой ребенком в его развитии в школе и дома. Возможности, предоставляемые машиной, и новые задачи образования неизбежно окажут заметное влияние на основные положения психологии развития, сложившиеся дидактические принципы и формы обучения»



Определение ОС

- *Операционная система* (ОС) — базовый набор функций, обеспечивающий интерфейс между пользователями (и приложениями) и аппаратурой компьютера

- *Легче сказать не что есть ОС, а для чего нужна и что она делает*



Три основные функции ОС

- Предоставление расширенной виртуальной машины, с которой удобнее работать, вместо реальной аппаратуры компьютера
- Повышение эффективности использования компьютера за счет рационального управления его ресурсами
- Организация безопасной деятельности пользователей и программ

Три основные функции ОС и …

Образно говоря, основной функцией операционной системы можно считать чародейство — превращение системы в нечто большее, чем есть на самом деле.

Например, операционная система может создать иллюзию одновременного исполнения нескольких программ на одном процессоре. В итоге пользователь воспринимает виртуальную машину как компьютер, имеющий архитектуру, отличную от реально существующей

С каких точек зрения можно рассматривать операционные системы?

В чем польза от изучения ОС?

Основные идеи, концепции и алгоритмы, лежащие в основе ОС, применимы ко многим другим областям программирования

ОС — большая и очень сложная программа, на примере которой можно изучать вопросы создания сложных программных продуктов

Изучение механизма и структуры операционных систем необходимо по многим причинам

СУБД

ПО для кластеров

ПО для грид

ПО для клаудов

Библиотеки

Компиляторы

Системы программирования

Популярные программные продукты могут рассматриваться как надстройки над ОС

И еще немного о пользе ОС

- Многие пользователи (и даже

Знай, читатель, что мудрость уменьшает жалобы, а не страдания!

Козьма Прутков

- П

Знай, программист, что понимание алгоритмов работы операционных систем облегчает страдания!

И.О. Одинцов

Почему так сложно спроектировать ОС?

- Почему никто до сих пор не сформулировал аналог закона Мура для ПО?
- Совершенствуются ли ОС с годами?
- А возможно ли разрабатывать огромные программы по гибким (agile) подходам?
- …

Три важных темы при изучении операционных систем

- Метафоричность
- Виртуализация
- Параллелизм

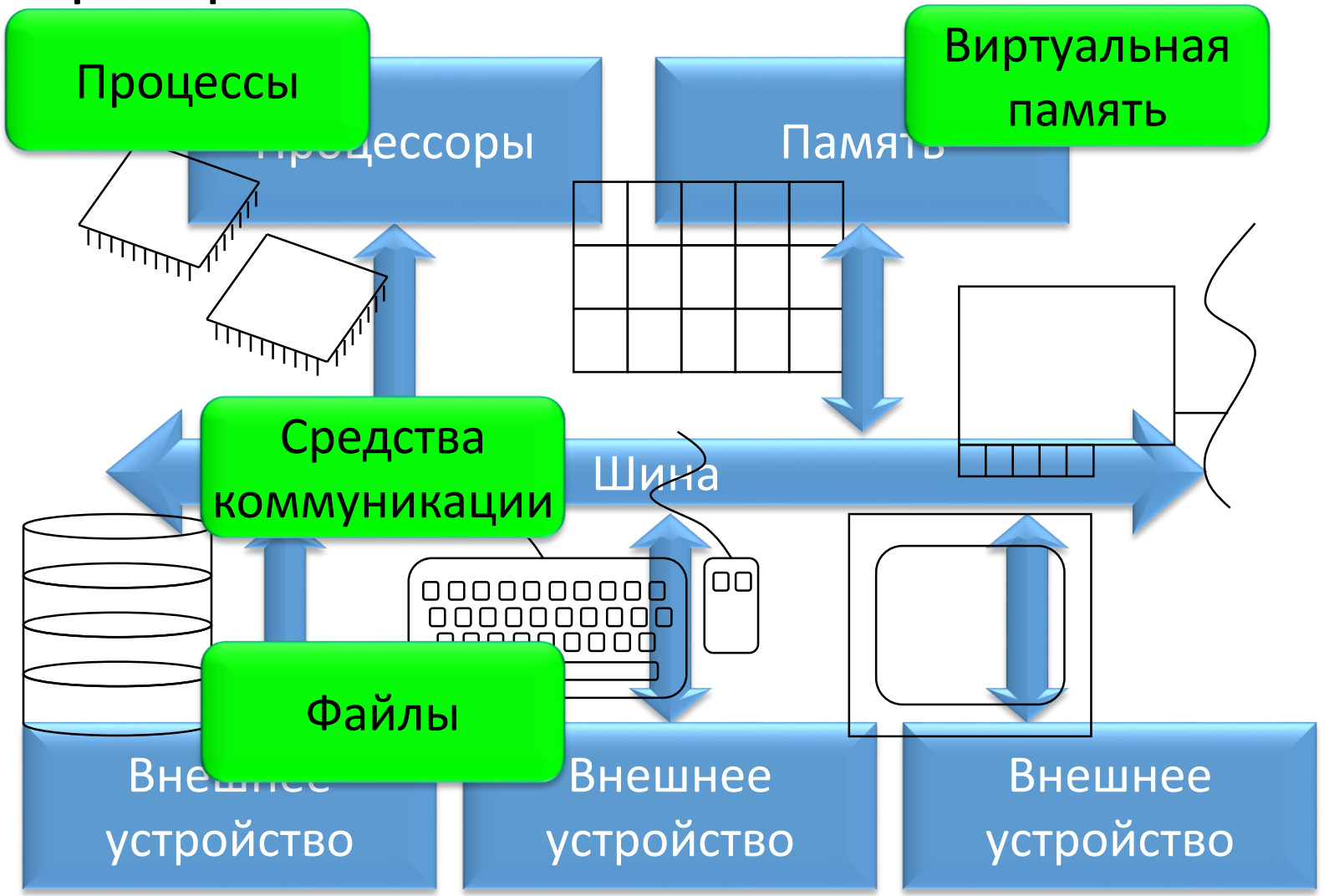
Метафоры

• Метафора – косвенное сообщение в виде истории или образного выражения, использующего сравнение
• Метафоры помогают найти наиболее подходящую аналогию, то есть соединить новый опыт с уже имеющимся
• Используйте метафору, чтобы объяснить что-то «бабушке»

Упражнение:
• «Что такое вычислительный грид?»
• «Что такое суперкомпьютер?»
• «Что такое БАК?»

Программа для детсадовцев: Горыныч онлайн

От аппаратных ресурсов к программным



Уточним – что такое процесс?

Несколько определений процесса

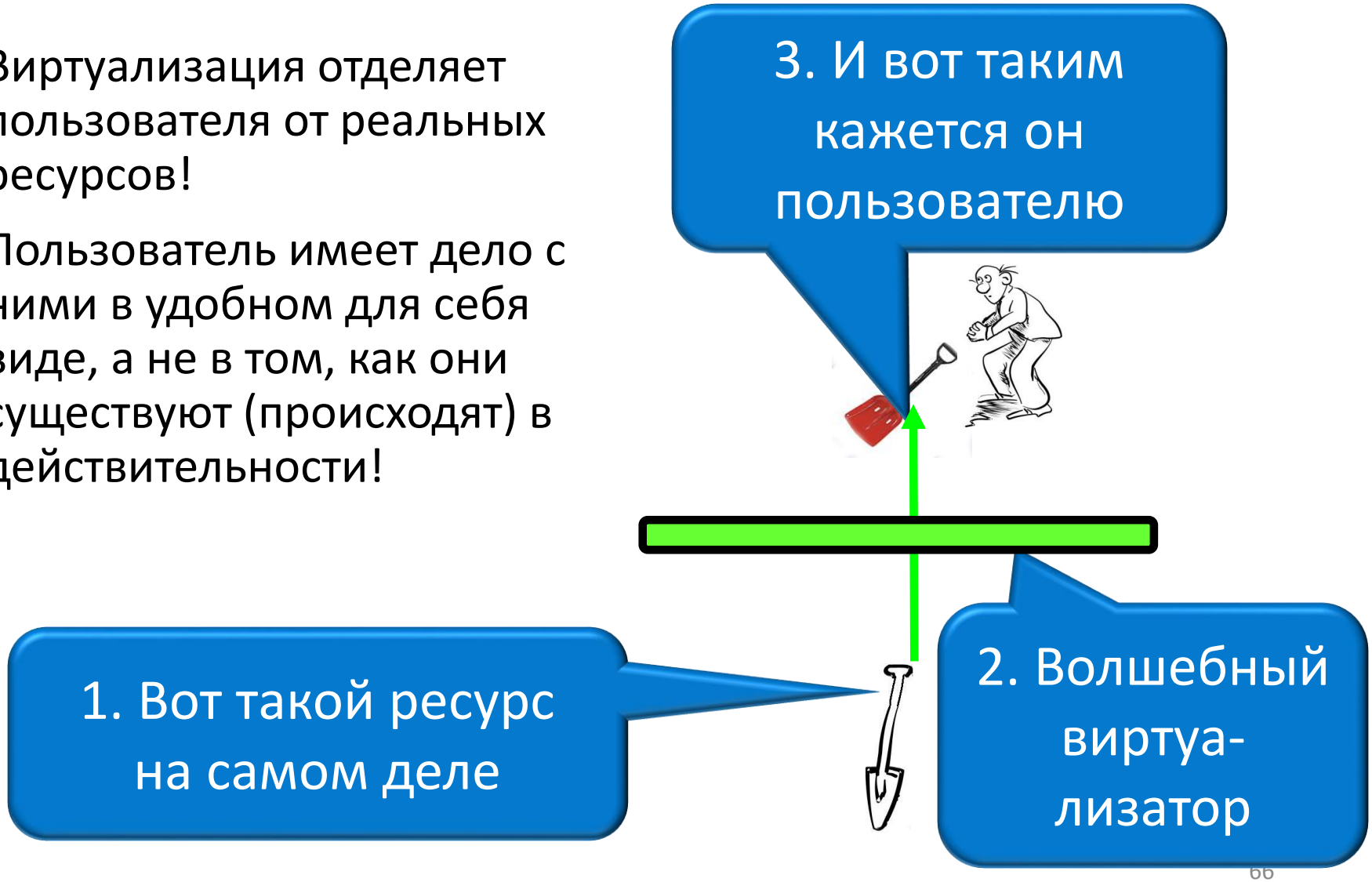
- *Процесс* — это абстракция, описывающая выполняющуюся программу
- *Процесс* — это исполнение последовательности действий в среде, включающей собственно выполняющуюся программу, а также связанные с ней данные и состояния (открытые файлы, текущий каталог и т. п.)
- С точки зрения операционной системы, *процесс* — это единица работы, заявка на потребление системных ресурсов
- С точки зрения аппаратной платформы, *процесс* — объект, которому выделяется процессор
- *Процесс* — это живая душа программы
- ...

«Одушевленность» процессов

Процессы как «жители коммунальной квартиры»:
переход из состояния в состояние,
борьба за ресурсы, …

Виртуализация

- Виртуализация отделяет пользователя от реальных ресурсов!
- Пользователь имеет дело с ними в удобном для себя виде, а не в том, как они существуют (происходят) в действительности!



Виртуализация (чудесное превращение реальности) в русских народных сказках

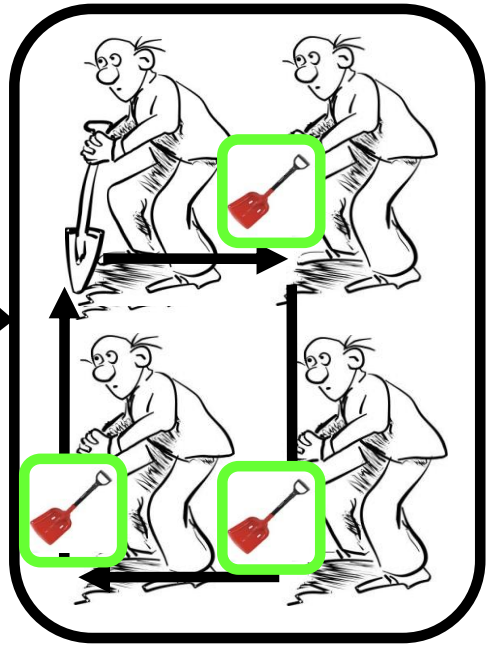
- Виртуализация видеоряда

- Виртуализация звукоряда

Виртуализация в операционных системах, на которых Вы работаете

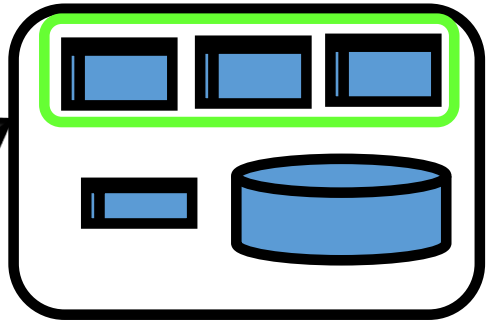
Вам кажется, что на Вашем однопроцессорном и одноядерном компьютере программы выполняются параллельно?

Нет! Процессор один. ОС пускает их ненадолго на процессор (ядро процессора) по очереди!

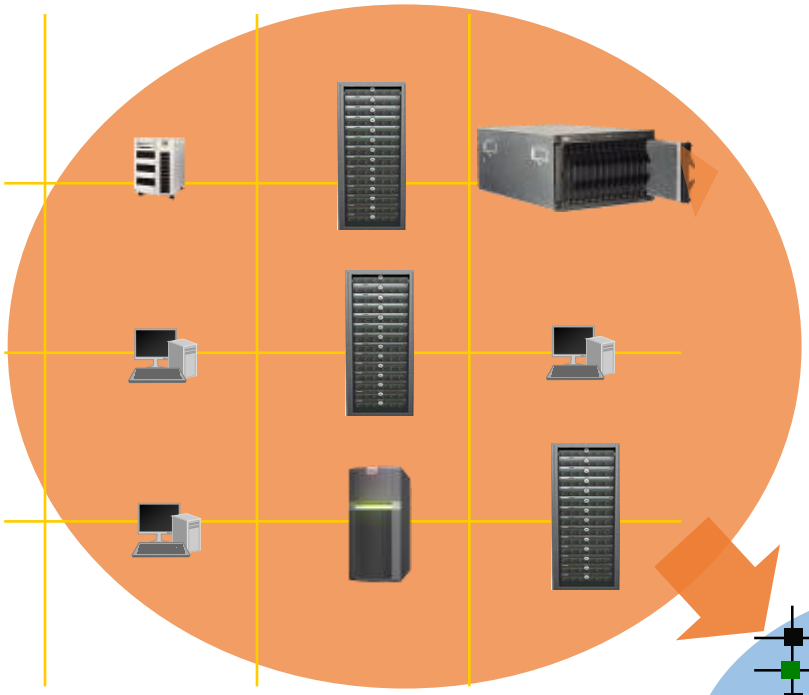


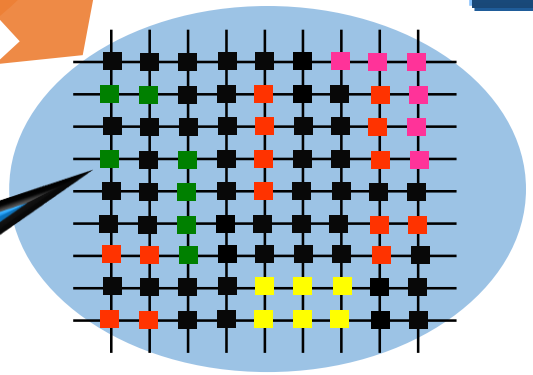
Вам кажется, что на Вашем компьютере все одновременно выполняющиеся программы находятся в оперативной памяти?

Нет! Оперативной памяти мало – она дорогая. ОС хранит их частично в оперативной памяти, а частично на дешевой дисковой памяти!



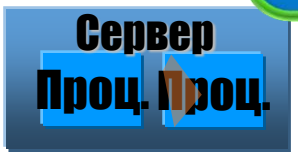
Виртуализация ресурсов



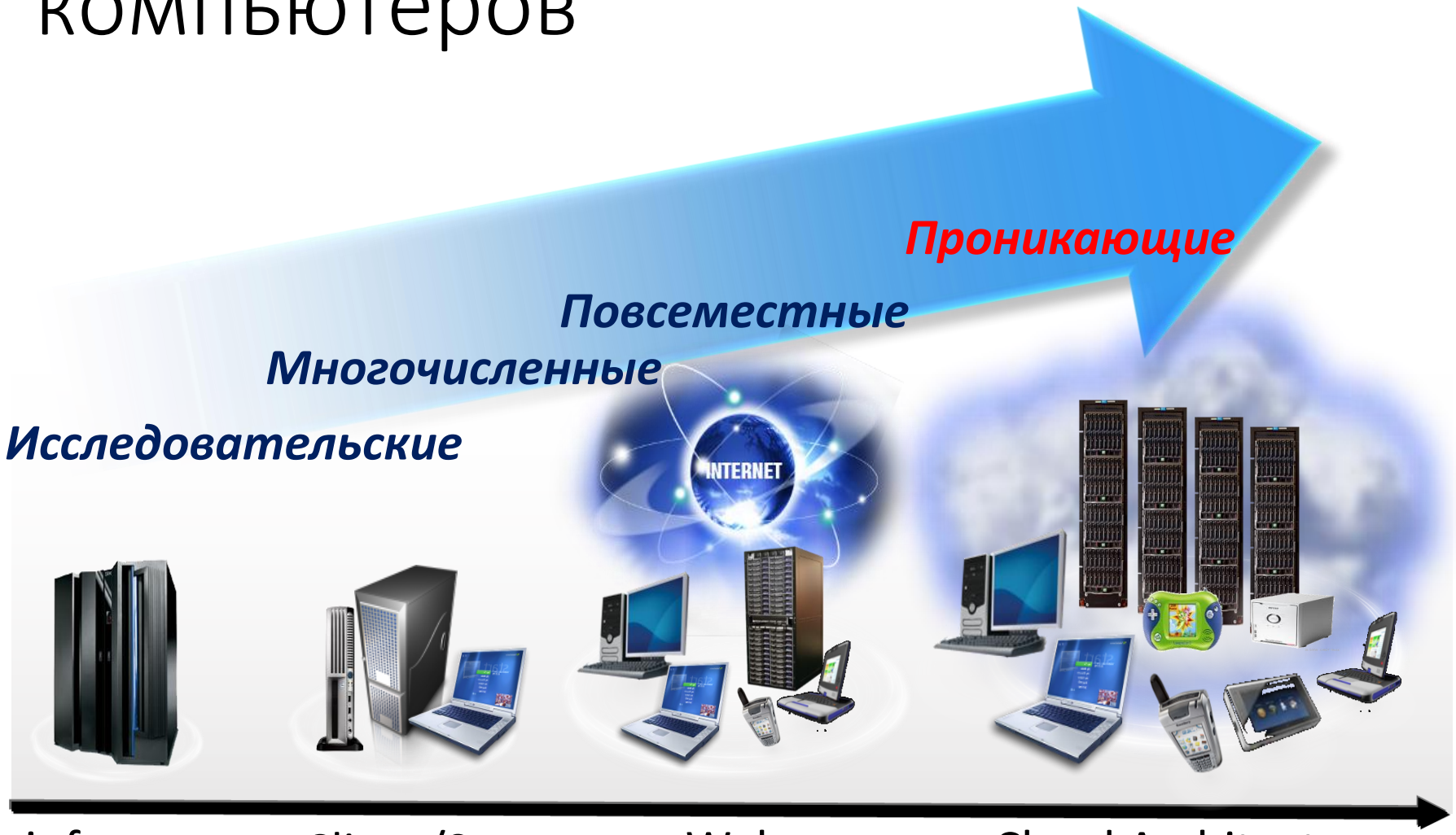


Это – грид ресурсов

Виртуальных машин может быть очень много

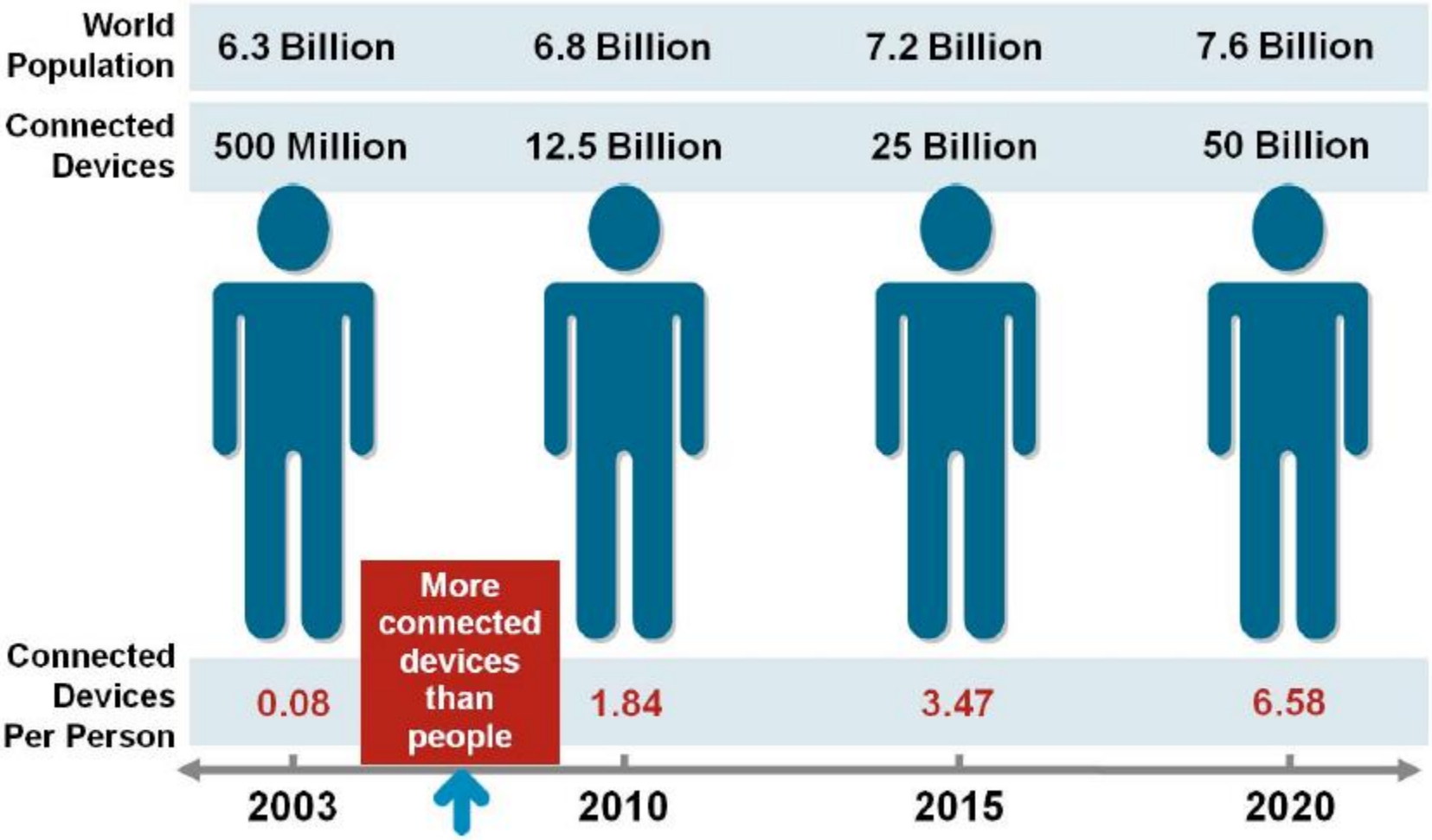


Эволюционная модель компьютеров



Уточним – что такое коммуникация процессов

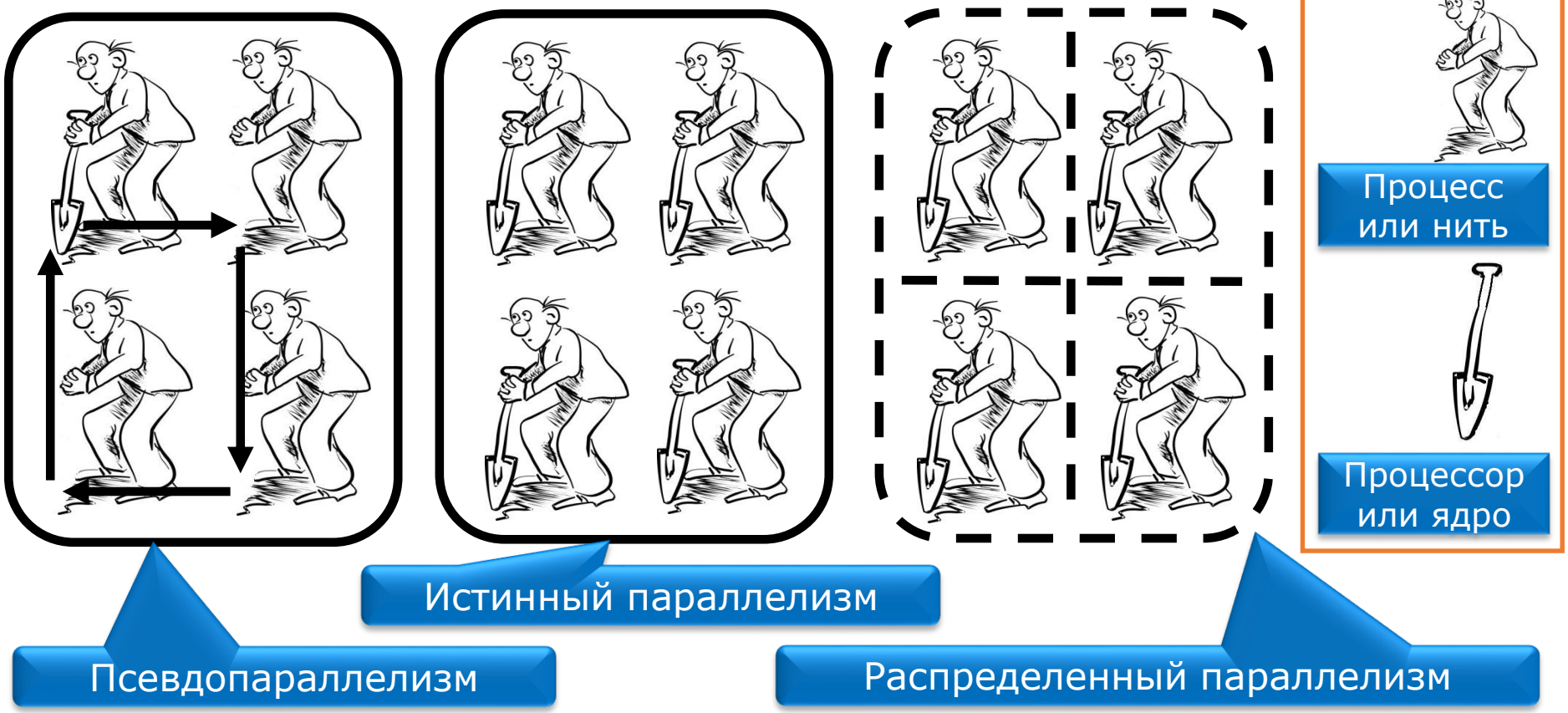
Интернет вещей!



Source: Cisco IBSG, April 2011

Уточним – какой бывает параллелизм

Параллелизм



- Процессы (нити) называются *параллельными*, если они выполняются одновременно. Они могут быть либо независимыми, либо взаимодействующими и нуждающимися в синхронизации

Задача №1: А в чем сложности написания параллельных программ?

- Пример: счетчик

Задача №2: И еще сложности написания параллельных программ

- Пример: холодильник

Семафоры

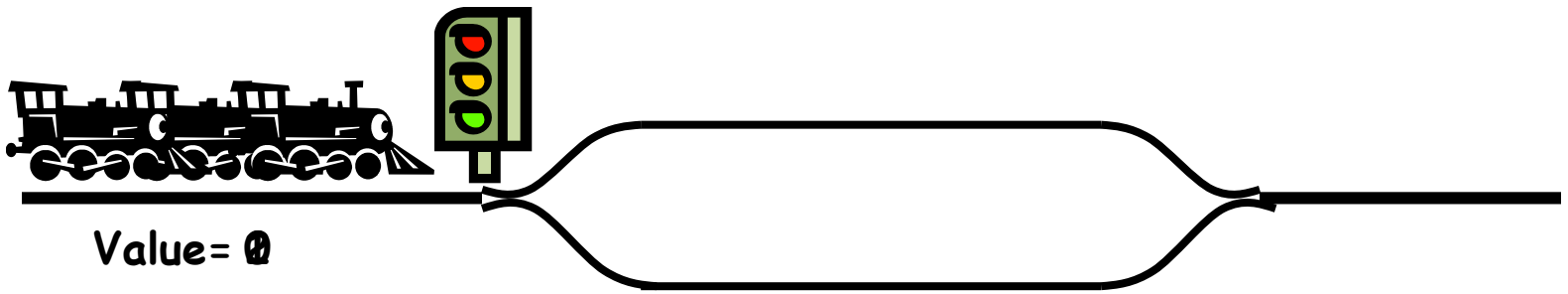
- Семафор — это защищенная переменная, значение которой можно запрашивать и менять только при помощи специальных операций P и V и при инициализации
- Концепция семафоров была предложена Дэйкстрой в начале 60-х годов XX века

P(S):

```
if ( S>0) then
    S:=S-1
else
    ожидать_в_очереди(S)
```

V(S):

```
if ( есть_процессы_в_очереди(S) ) then
    одному_продолжить (S)
else
    S:=S+1
```



Задача №3: И опять сложности написания параллельных программ

- Пример: круговое ожидание

Многопоточность – это сложно

«Кто виноват?»

- ☑ Слишком низкий уровень абстракции
- ☑ Отсутствие необходимых знаний и опыта
- ☑ Некоторые концепции **действительно** сложны!

«Что делать?»

- ☐ Нанять эксперта в разработке многопоточных программ
- ☐ Стать таким экспертом
- ☑ Использовать другие подходы к параллелизму

Будьте экспертом в своей области!

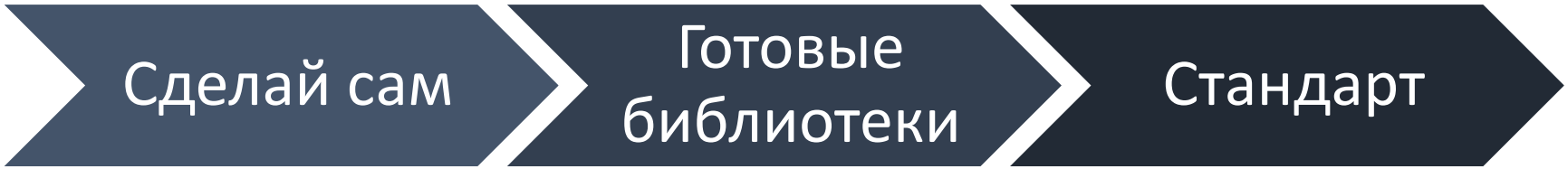
Deja vu

Мы это всё уже проходили, и не раз

- В 1990-х: «оконный» интерфейс для DOS, класс string и т.д.
- До появления STL: списки, очереди, ... и использующие их алгоритмы

Не надо «изобретать велосипед»!

- Используйте C++-библиотеки «параллельных шаблонов» /*parallel patterns*/
- Есть выбор: от Intel, Microsoft, NVidia, AMD, Qualcomm, …
- Многие с открытым исходным кодом

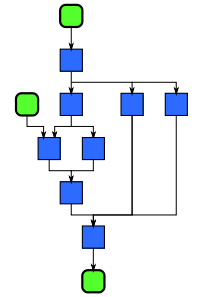


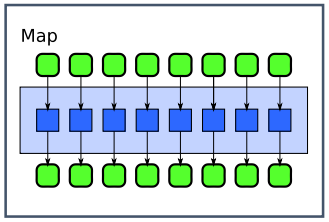
Один из возможных профессиональных подходов к параллелизму

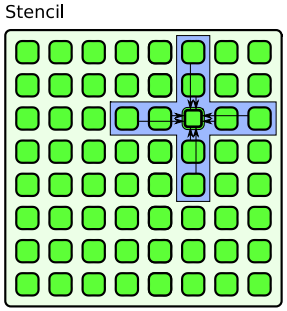
- Определите параллелизм на алгоритмическом уровне
  - Разбейте алгоритм на отдельные вычислительные блоки
  - Определите зависимости между блоками
- Найдите подходящий параллельный шаблон
- Примените этот шаблон с помощью выбранной библиотеки
- Если нужного шаблона не нашлось, но есть API для использования «задач», попробуйте применить его
- Оставьте библиотеке сложную и рутинную работу

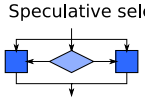
Параллельное программирование может быть доступным

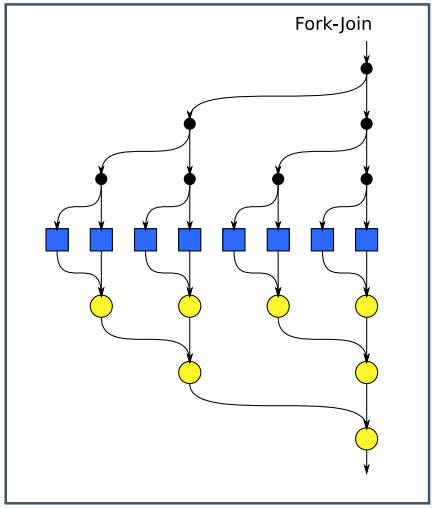
Параллельные шаблоны

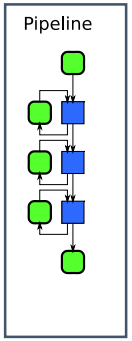


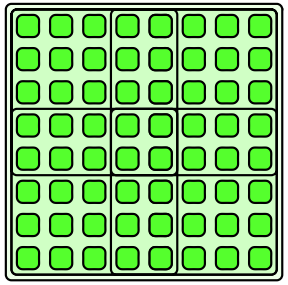


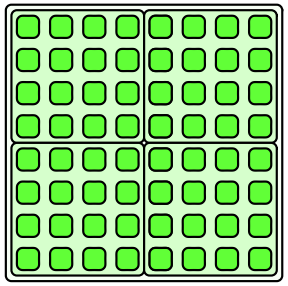


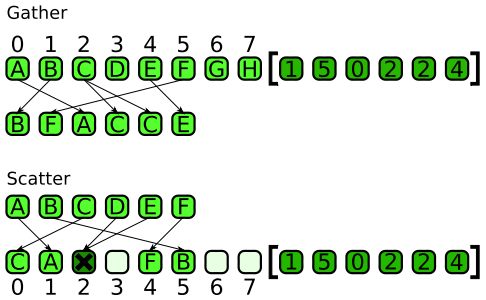


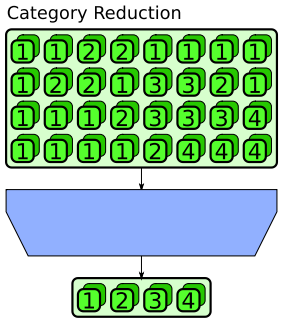


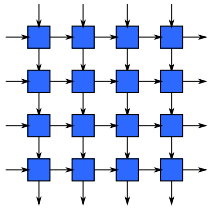


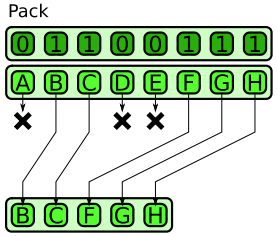


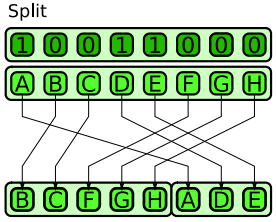


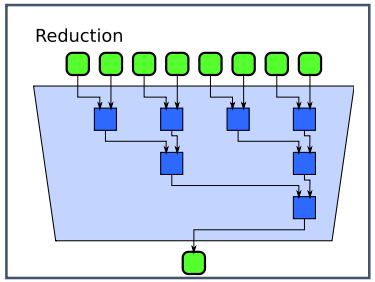


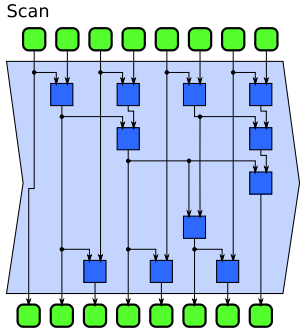


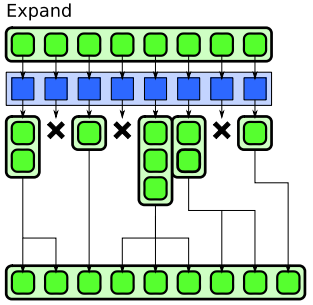












Учиться! Учиться! Учиться!
Параллельным вычислениям!