Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ (лекция 7)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2020

Векторно-конвейерные компьютеры

Векторно-конвейерные компьютеры

Функциональное устройство:

- *скалярное;*
- *конвейерное (операция делится на несколько микроопераций, количество микроопераций определяет число ступеней конвейера).*

Команда:

- *скалярная (все аргументы - скаляры);*
- *векторная (например, сложить все элементы массива x с числом b).*

Компьютер:

- *скалярный;*
- *векторный;*
- *конвейерный.*

Векторно-конвейерные компьютеры

(векторизация, векторы данных, независимые операции)

Процесс поиска подходящих фрагментов в программе и их замена векторными командами - ***векторизация программы****.*

Пример векторизуемого фрагмента:
for(i=0; i<n; i++) c[i] = a[i]+b[i];

Компилятор сгенерирует последовательность векторных команд:

- *загрузка векторов a и b из памяти в векторные регистры,*
- *векторная операция сложения,*
- *запись содержимого векторного регистра в память.*

Если весь фрагмент программы удалось заменить векторными командами, то говорят о его ***полной векторизации****. В противном случае мы имеем дело с частичной векторизацией или невозможностью векторизации фрагмента вовсе.*

Истоки векторности: *линейная организация памяти, использование массивов.*

Векторно-конвейерные компьютеры

(векторизация, векторы данных, независимые операции)

Условия векторизации:

- *наличие векторов-аргументов;*
- *над всеми элементами векторов должны выполняться одинаковые, независимые операции, для которых существуют аналогичные векторные команды в системе команд компьютера.*

Вектор *- упорядоченный набор однотипных данных, все элементы которого размещены в памяти компьютера с одинаковым смещением друг относительно друга.*

Простейшие примеры векторов:

- *одномерные массивы;*
- *строки и столбцы матриц;*
- *диагональ квадратной матрицы;*
- *многомерный массив целиком.*

Не вектор - поддиагональная часть двумерной матрицы.

Векторно-конвейерные компьютеры

(векторизация, векторы данных, независимые операции)

В векторизуемом фрагменте могут использоваться и простые переменные.

for(i=0; i<n; i++) b[i] = a[i]+s;

*Основные кандидаты для векторизации - **самые внутренние циклы** всех циклических конструкций программы.*

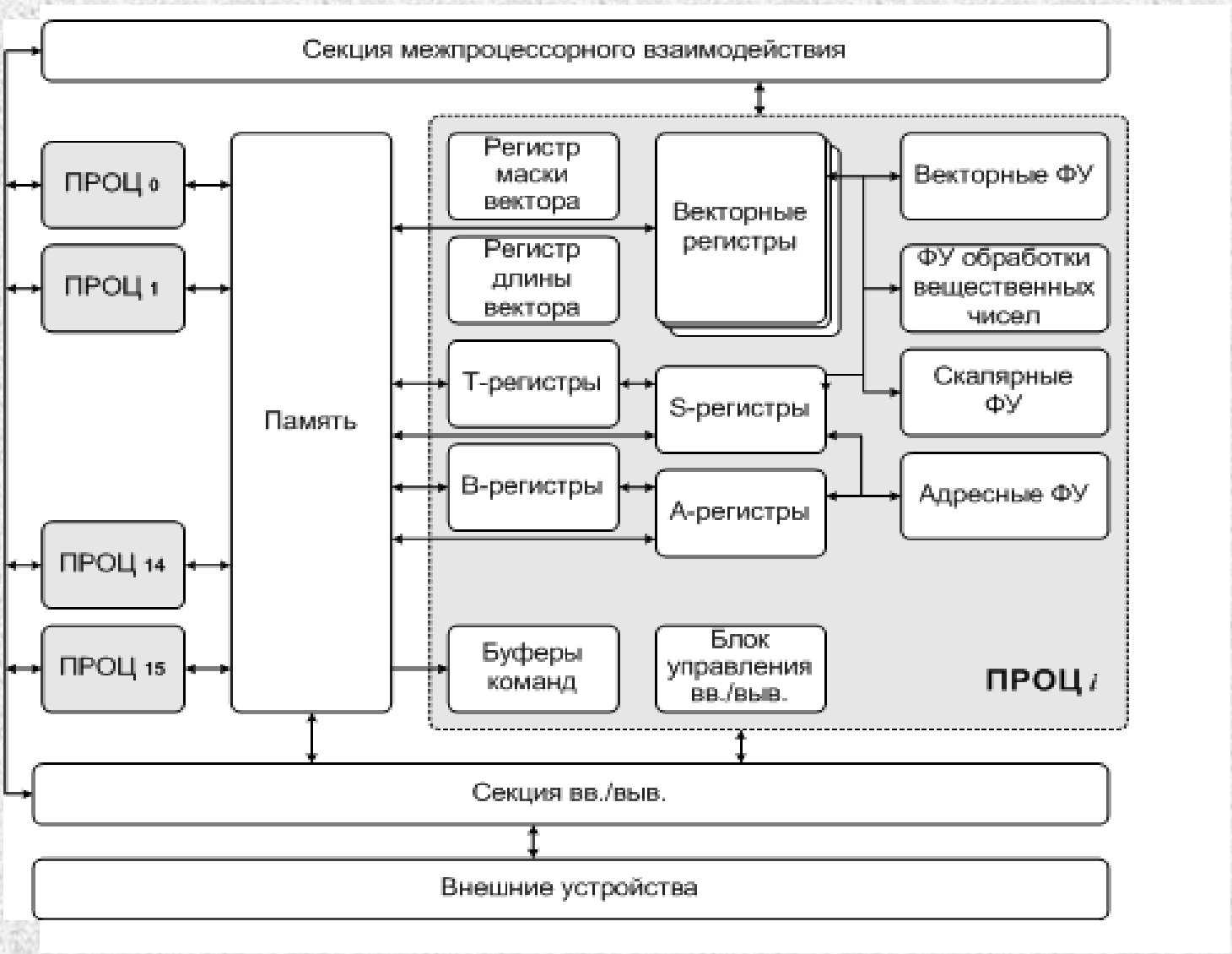
for(i=0; i<n; i++) a[i] = a[i-1]+b[i];

*Необходимо **отсутствие информационной зависимости** между операциями!*

for(i=0; i<n; i++) a[i] = f(a[i], b[i]);

Неизвестно, какая операция соответствует вызову функции f, и, следовательно, на какие векторные команды можно заменить данный фрагмент.

Векторно-конвейерные компьютеры



Общая схема суперкомпьютера CRAY C90 (1991 г.)

Функциональные устройства

- *Все функциональные устройства конвейерные.*

- *Число ступеней различно, однако каждая ступень каждого устройства всегда срабатывает за один такт.*

- *Все функциональные устройства независимы и могут работать одновременно друг с другом.*

- *Функциональные устройства данного компьютера делятся на четыре группы:*
  - ✓*адресные – операции над 32-разрядными целыми числами;*
  - ✓*скалярные – операции над 64-разрядными целыми числами;*
  - ✓*для векторных команд – операции над целочисленными векторами;*
  - ✓*функциональные устройства для выполнения вещественной арифметики – операции над 64-разрядными числами в форме с плавающей запятой (как в векторном, так и в скалярном режиме).*

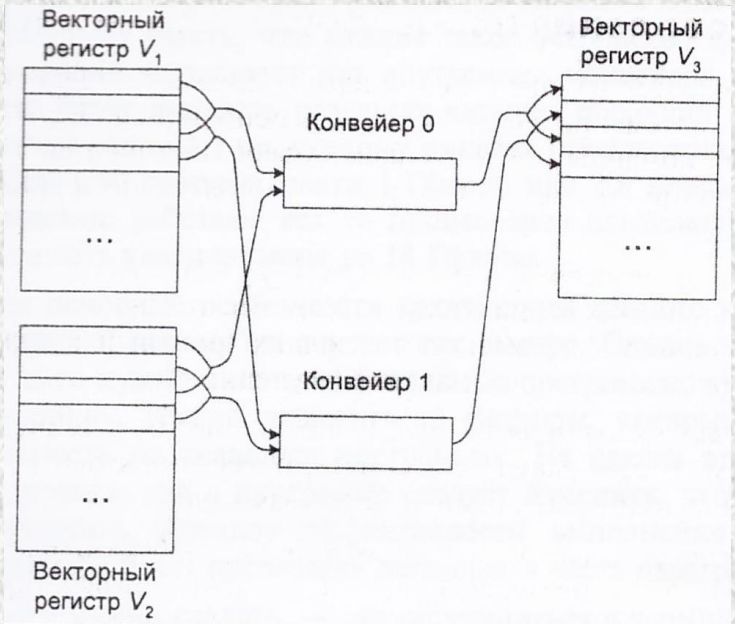
Время разгона конвейера и секционирование векторных команд

```
for(i = 0; i < n; i++)
    a[i] = b[i]*s+c[i];
```

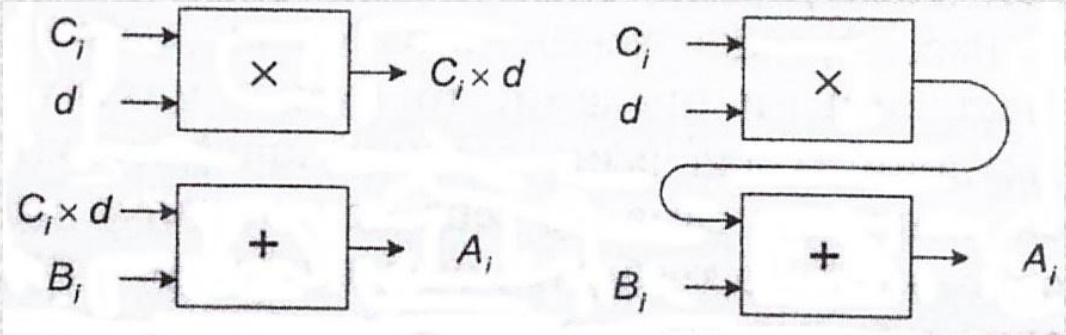
Длина вектора	Производительность (Mflops)
1	7.0
4	27.6
32	181.9
128	433.7
129	364.3
256	548.0
257	491.0
8192	802.0

Производительность CRAY C90 на операции $a_i = b_i*s+c_i$.

Дублирование и зацепление векторных устройств



Дублирование векторных устройств и устройств для вещественной арифметики



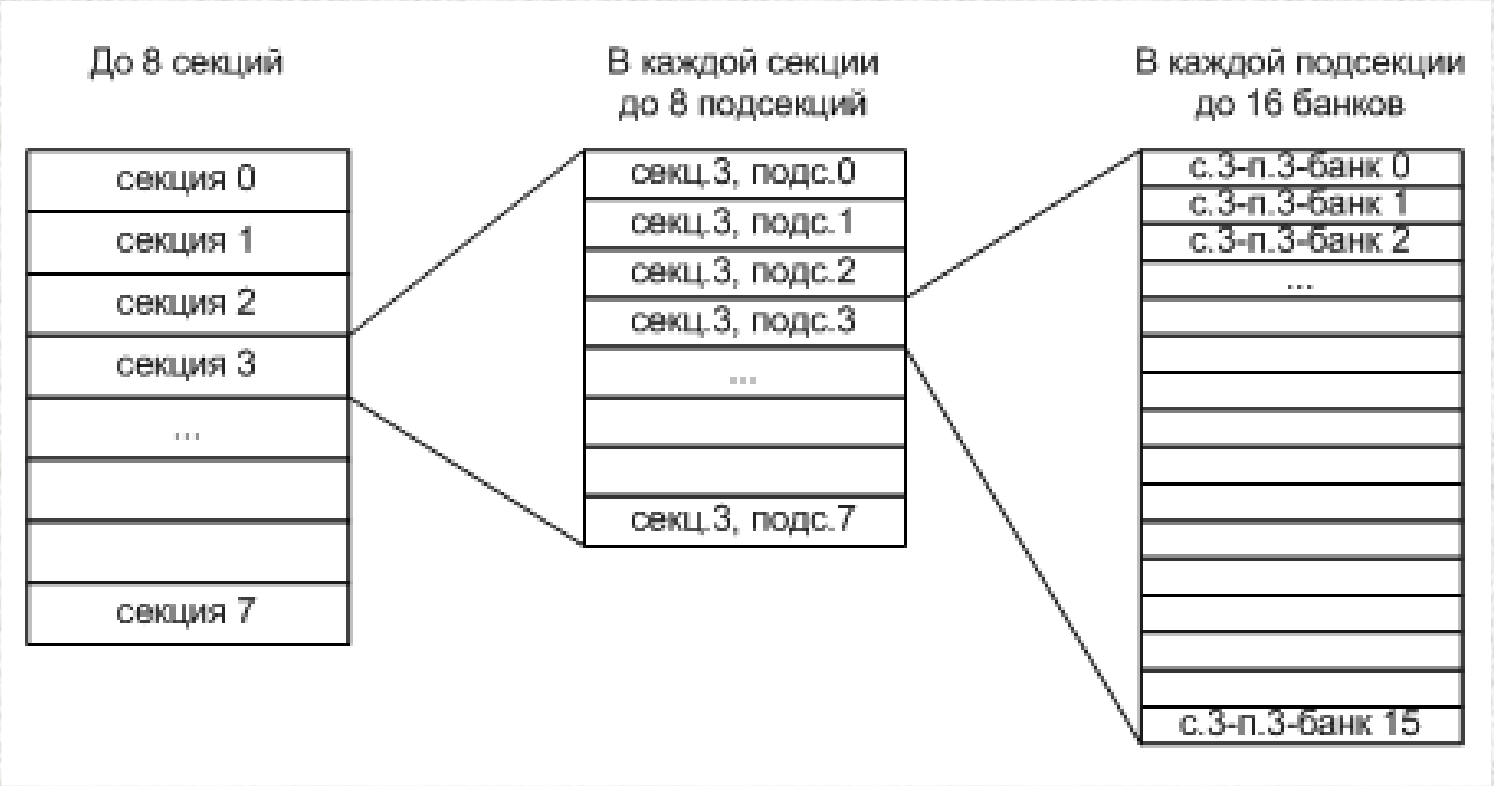
Зацепление векторных операций:

x - l_1 ступеней
+ - l_2 ступеней

- *Без зацепления: (l_1+n–1) + (l_2+n–1) = l_1+l_2+2*n–2 тактов*

- *С зацеплением: l_1+l_2+n–1 тактов*

Конфликты в памяти



- *При одновременном обращении к одной и той же секции возникает конфликт, который разрешается за 1 такт. В этом случае один из запросов продолжает обрабатываться, а другой просто блокируется на один такт.*
- *Если происходит одновременное обращение к одной и той же подсекции одной секции, то время на разрешение конфликта уже может достигать 6 тактов.*

Конфликты в памяти

Последовательные адреса идут с чередованием :

адрес 0 — в 0-й секции, 0-й подсекции, 0-м банке;
адрес 1 — в 1-й секции, 0-й подсекции, 0-м банке;
адрес 2 — в 2-й секции, 0-й подсекции, 0-м банке;
...
адрес 8 — в 0-й секции, 1-й подсекции, 0-м банке;
адрес 9 — в 1-й секции, 1-й подсекции, 0-м банке;
...
адрес 63 — в 7-й секции, 7-й подсекции, 0-м банке;
адрес 64 — в 0-й секции, 0-й подсекции, 1-м банке;
адрес 65 — в 1-й секции, 0-й подсекции, 1-м банке;
...

Максимальное число конфликтов будет происходить при постоянном обращении к одной и той же подсекции одной и той же секции. Это заведомо произойдет, например, при выполнении процессором векторной операции чтения данных, расположенных с шагом, кратным 64.

Конфликты в памяти

- *Операции чтения/записи последовательно расположенных данных проходят без возникновения конфликтов. В частности, все одномерные массивы будут обрабатываться именно так.*

- *При выборке данных с любым нечетным шагом конфликтов также не возникнет.*

- *Чем б**о**льшая степень двойки присутствует в качестве сомножителя в шаге выборки данных, тем больше времени требуется на разрешение возникающих конфликтов.*

Влияние конфликтов в памяти на производительность

```
for(i = 0; i < n*k; i +=k)
  a[i] = b[i]+c[i]*d;
```

Шаг по памяти	Производительность (Mflops)
1	705.2
2	444.6
4	274.6
8	142.8
16	84.5
32	44.3
64	22.7
128	22.6

Производительность CRAY C90 на операции $a_i = b_i + c_i * d$, n=1000

Влияние конфликтов в памяти на производительность
(многообразие проявления конфликтов)

```
Do i = 1, n
   Do j = 1, n
      Do k = 1, n
         X(i,j,k) = X(i,j,k) + P(k,i)* Y(k,j)
```

X(N1,N2,N3)

Фортран: хранение массивов “по столбцам”:

X(i,j,k) ≈ X(i+1,j,k) расстояние 1,

X(i,j,k) ≈ X(i,j+1,k) расстояние N1,

X(i,j,k) ≈ X(i,j,k+1) расстояние N1*N2.

Реальное описание массива: X(40,40,N)

X(i,j,k) X(i,j,k+1) расстояние 40*40 = 1600 = 64*25.

Решение? Описание массива X(41,41,N)

Каналы процессор-память

Три канала передачи данных, два из которых могут работать на чтение из памяти, а третий на запись.

Длина вектора	Производительность (Mflops)
10	57.0
100	278.3
1000	435.3
12801	445.0

Производительность CRAY C90 на операции

$a_i = b_i * c_i + d_i$

Операции чтения/записи в векторные регистры, ограниченное число векторных регистров

Раскрутка цикла:

```
for(j = 1; j <= 120; j++)
    for(i = 1; i <= n; i++)
        d[i] = d[i]+s*p[i][j-1]+t*p[i][j];
```

*120*3=360 операций чтения*
120 операций записи

```
for(j = 1; j <= 120; j+=2)
    for(i = 1; i <= n; i++)
        d[i] = d[i]+s*p[i][j-1]+t*p[i][j]+
                   s*p[i][j]+t*p[i][j+1];
```

*60*4=240 операций чтения*
60 операций записи

Глубина раскрутки	Производительность (Mflops)
1	612.9
2	731.6
3	780.7
4	807.7

Зависимость производительности CRAY C90 от глубины раскрутки

Несбалансированное использование устройств, отсутствие операции деления

Длина вектора	Производительность (Mflops)			
	$a_i=b_i+c_i$	$a_i=b_i/c_i$	$a_i=s/b_i+t$	$a_i=s/b_i\times t$
10	35.5	24.8	49.7	46.1
100	202.9	88.4	197.4	166.5
1000	343.8	117.2	283.8	215.9

Производительность CRAY C90 на различных операциях

Пользователь: почему?

$A_{ijk} = A_{i-1jk} + B_{jk} + B_{jk}$, i=1,40; j=1,40; k=1,1000

Cray C90, пиковая производительность 960 Mflop/s

```
do k = 1, 1000
   do j = 1, 40
      do i = 1, 40
         A(i,j,k) = A(i-1,j,k)+B(j,k)+B(j,k)
```

Производительность: **20** Mflop/s на Cray C90

Пользователь: почему?

$A_{ijk} = A_{i-1jk} + B_{jk} + B_{jk}$, i=1,40; j=1,40; k=1,1000

Cray C90, пиковая производительность 960 Mflop/s

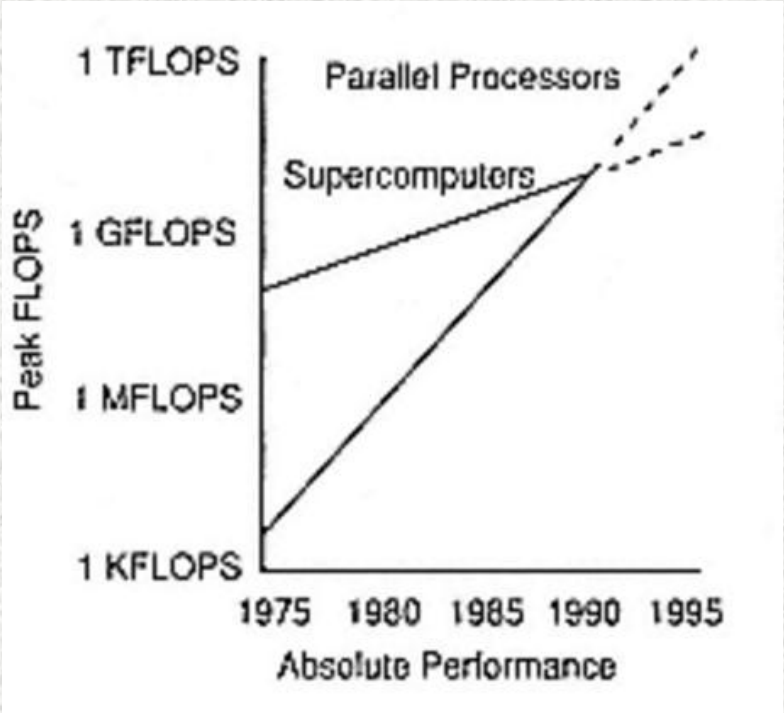
```
do i = 1, 40, 2
   do j = 1, 40
      do k = 1, 1000
         A(i,j,k) = A(i-1,j,k)+2*B(j,k)
         A(i+1,j,k) = A(i,j,k)+2*B(j,k)
```

Производительность: **700** Mflop/s на Cray C90

Векторно-конвейерные компьютеры

- Предсказывалось, что векторные компьютеры устареют к 1990 году.
- Основная причина – быстрый прогресс в производительности микропроцессоров. Векторные инструкции в микропроцессорах не использовались.

- Что ещё привело к такому положению дел?
  I. Сокращение государственных расходов США на суперкомпьютеры (после 1989 года).
  II. Распространение микропроцессоров Intel и AMD -> большие объёмы продаж -> много средств на исследование и разработку.



Векторно-конвейерные компьютеры

- Возвращение векторности по причине эволюции микропроцессоров и появления в них векторных регистров.
- MMX Instruction set (1997) – 8 64-битных регистров, только операции с целыми числами.
- SSE (Streaming SIMD Extension, 1999) – 8 128-битных регистров, операции с плавающей точкой. Получила развитие в виде SSE2, SSE3 и.т.д. (добавление новых инструкций).
- AVX (2008) – 16 256-битных регистров, *инструкции с 3 операндами*:

a = a + b -> c = a + b.

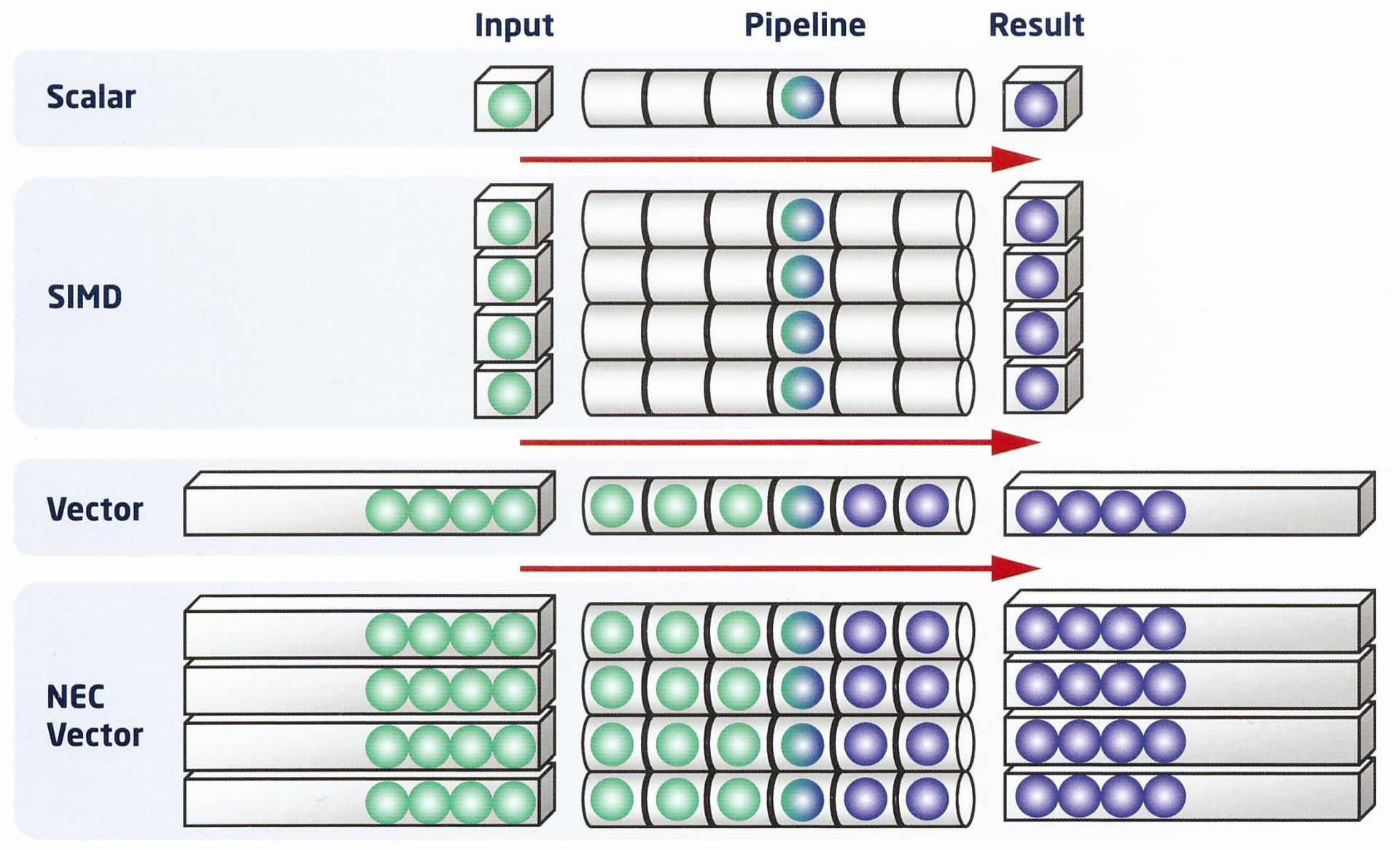
- AVX2 (2013) – имеются специальные инструкции для целых чисел. Добавлены инструции для операций вида a = b + c * d.
- AVX-512 – 32 512-битных регистра. Поддерживается в Xeon Phi KNC, KNL и Intel Skylake-X.

Векторно-конвейерные компьютеры

- AltiVec / Velocity Engine / VMX – в своё время прямой конкурент SSE.
  I. Регистры можно загружать и сохранять только в память.
  II. Больший (по сравнению с SSE) набор допустимых типов данных и операций.
  III. 32 128-битных регистра, инструкции с тремя операндами.
- ARM SVE – расширение для набора инструкций A64.
  I. Длина вектора от 128 до 2048 бит, кратна 128. Предпочитаемой длины вектора нет, всё зависит от конкретного «железа».
  II. Приложения подстраиваются под имеющийся объём.
  III. Не нужно ничего перекомпилировать / переписывать.
  IV. Ориентация на борьбу с проблемами автовекторизации (современные компиляторы часто не в состоянии провести векторизацию из-за зависимостей внутри вектора).
  V. Предназначен для решения научных задач с помощью HPC.

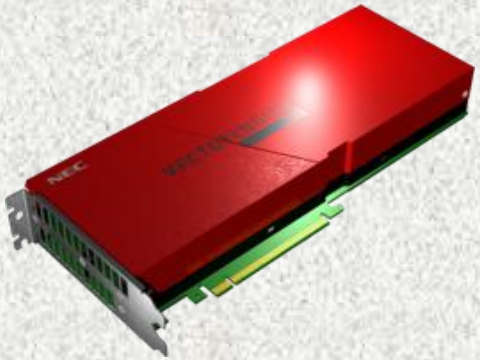
NEC SX-Aurora TSUBASA

NEC SX-Aurora TSUBASA

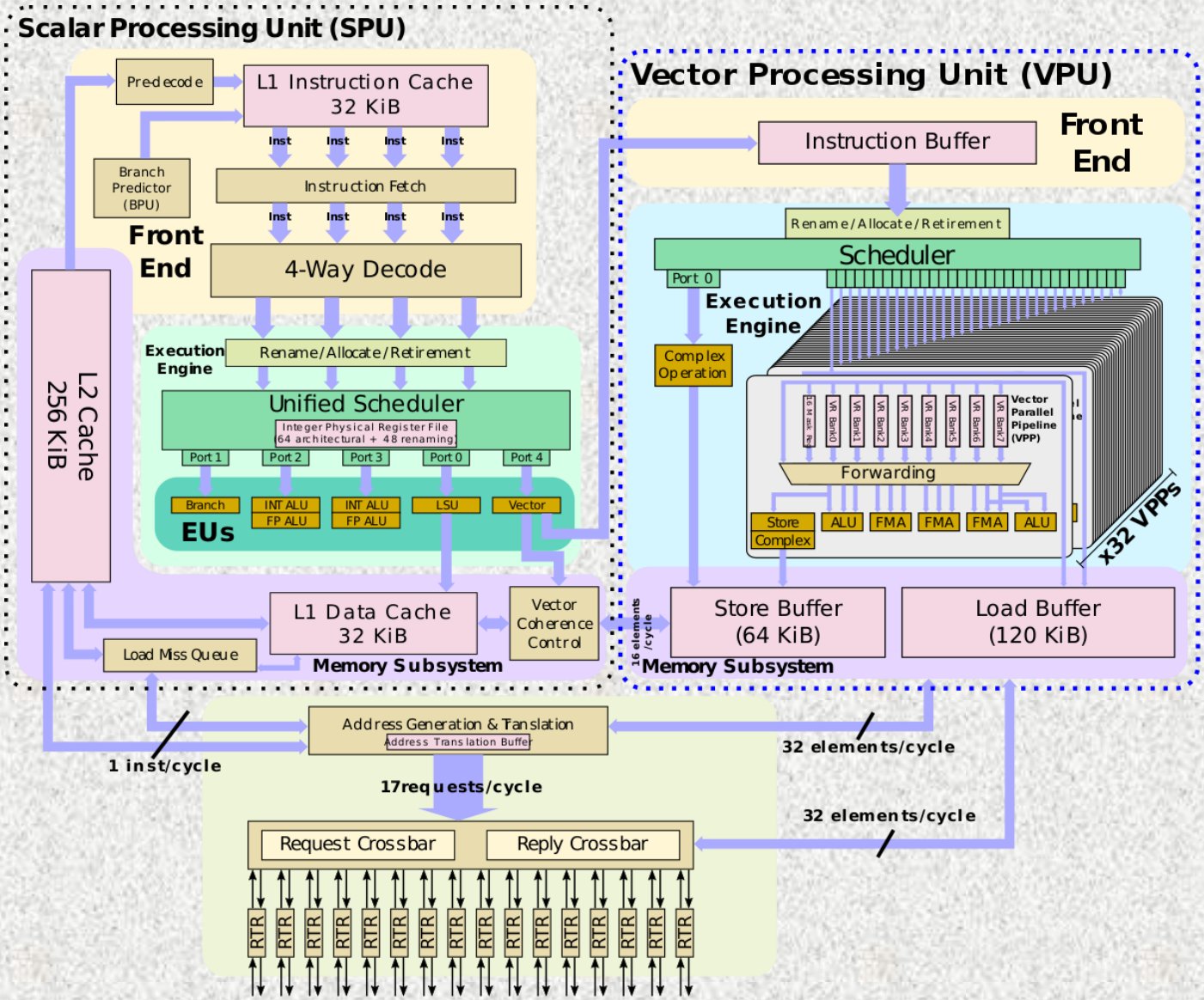


Совмещение векторных и SIMD команд

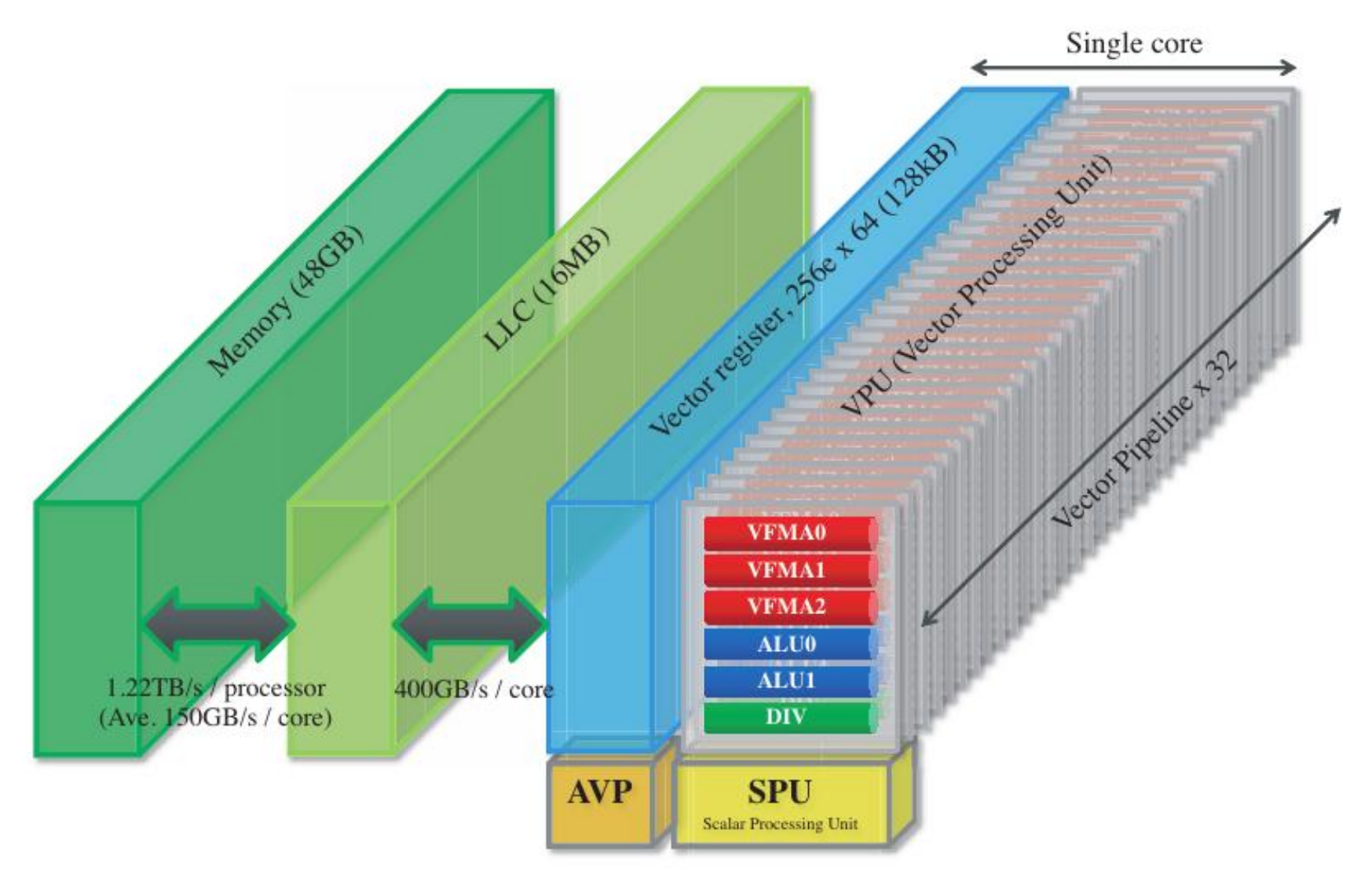
NEC SX-Aurora TSUBASA



Vector Engine (VE) в виде карты PCIe включает в себя 8 ядер, состоящих из скалярного и векторного блоков

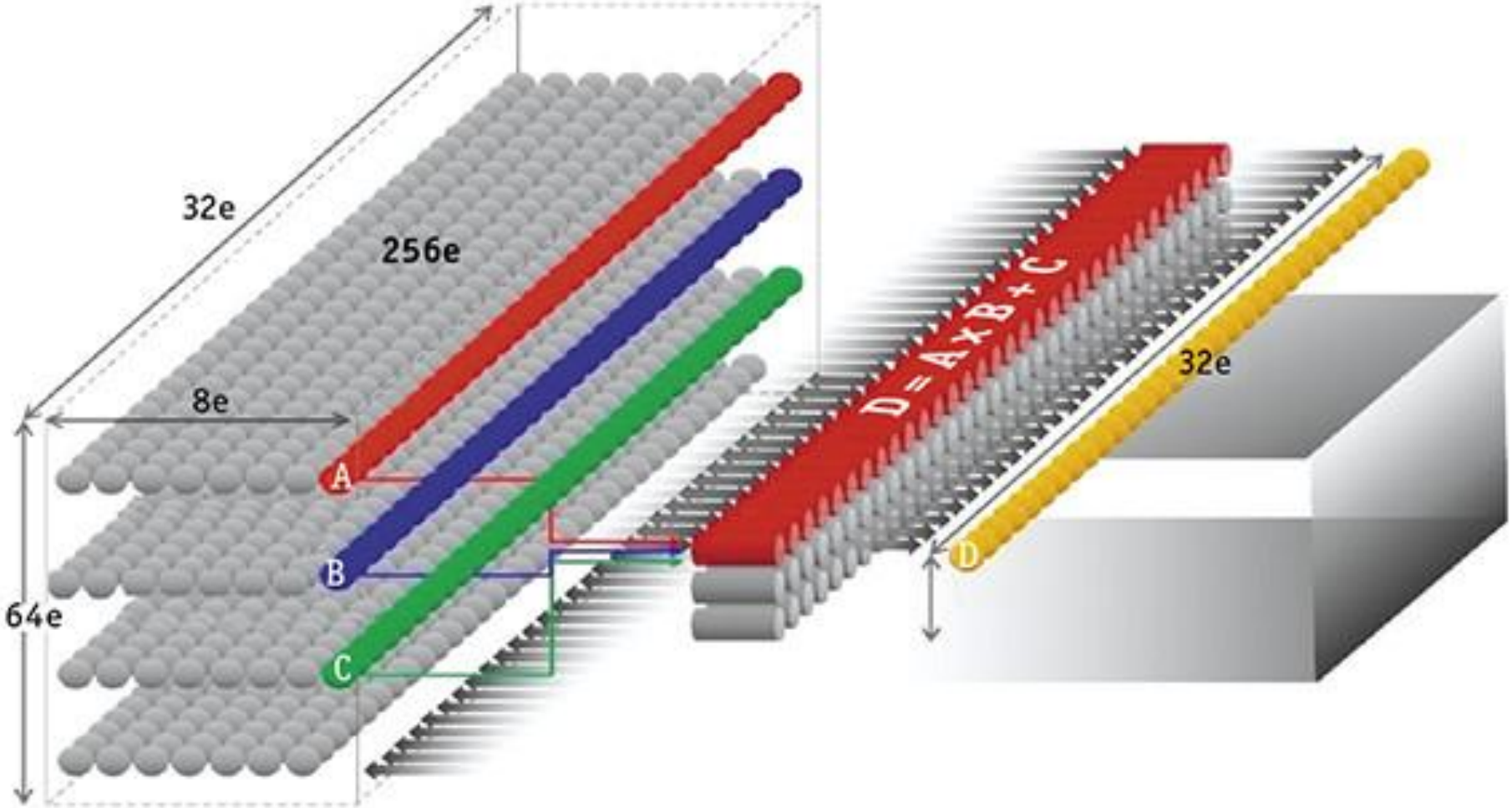


NEC SX-Aurora TSUBASA



В каждом ядре содержится по 64 векторных регистра длиной по 256 64-разрядных элементов. Суммарная ёмкость регистров в составляет 128 Кбайт.

NEC SX-Aurora TSUBASA



3 устройства FMA (Fused Multiply Add) выполняют операцию a * b + c над векторными регистрами – каждый такт 32 результата. Суммарно 32x3x2=192 Flop/такт. При тактовой частоте 1.6 ГГц получаем 307.2 GFlop/s на ядро или 2.4576 TFlop/s на процессор.

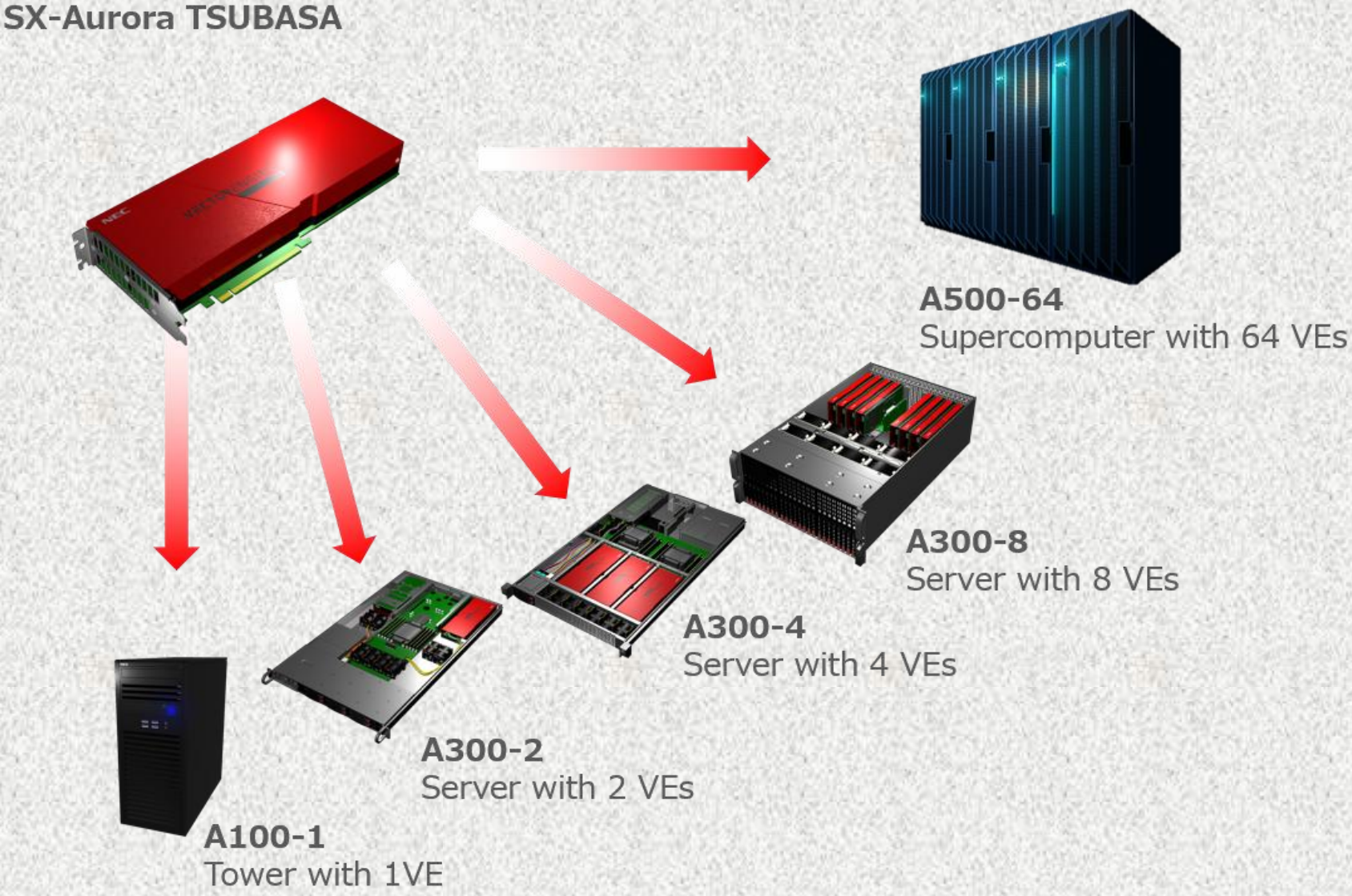
NEC SX-Aurora TSUBASA

В отличие от ускорителей, программа целиком выполняется на VE. Vector Host (VH) используется для компиляции приложений и реализации фонкций ОС, таких как выделение ресурсов, взаимодействие с файловой системой и т.д.

В рамках одного узла разные VE могут связываться друг с другом через PCIe. Большие параллельные системы, созданные с помощью SX-Aurora, используют в качестве коммуникационной сети Infiniband.

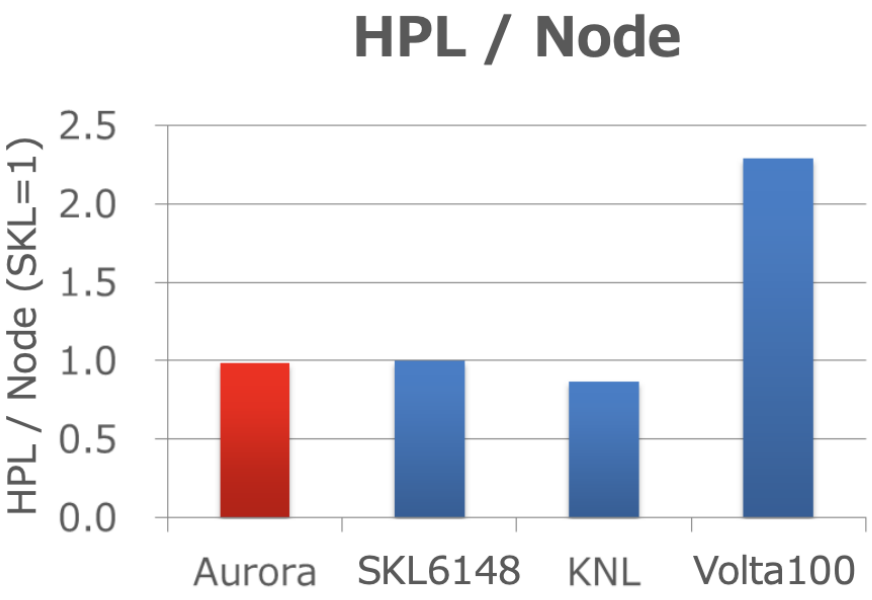
Операционная система VE называется VEOS и полностью выгружена в хост-систему, векторный хост (VH).

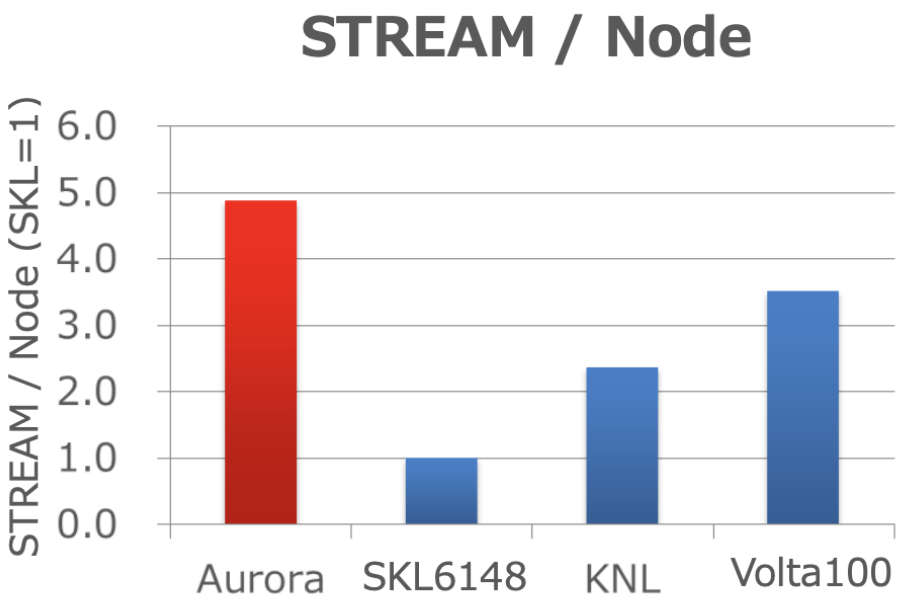
NEC SX-Aurora TSUBASA



Диапазон платформ TSUBASA от рабочих станций A100 до стоечных суперкомпьютеров A500 с пиковой производительностью 157.3 TFlop/s.

NEC SX-Aurora TSUBASA

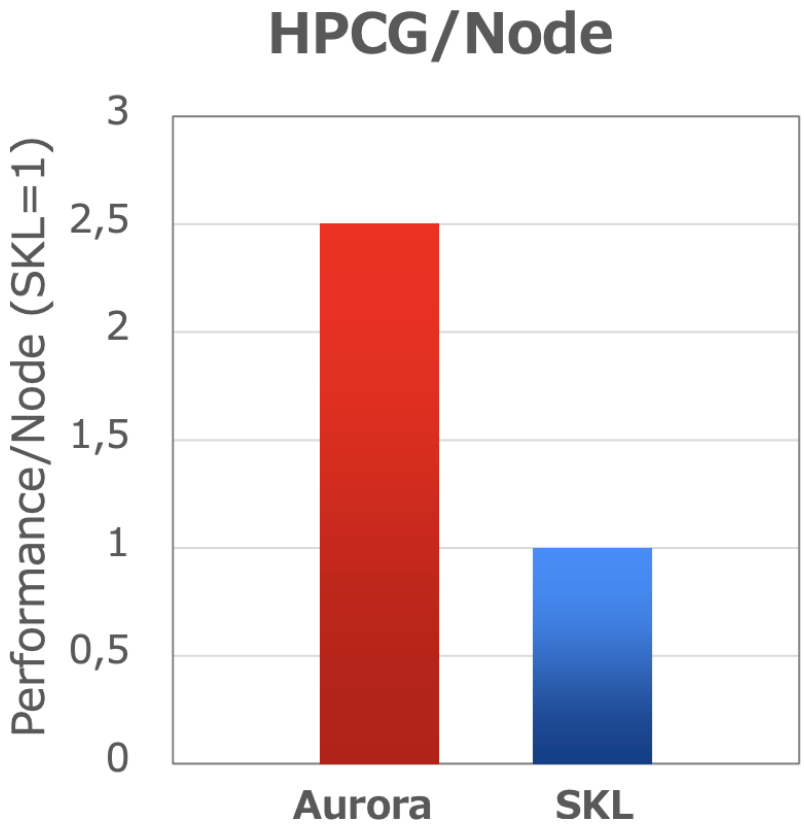


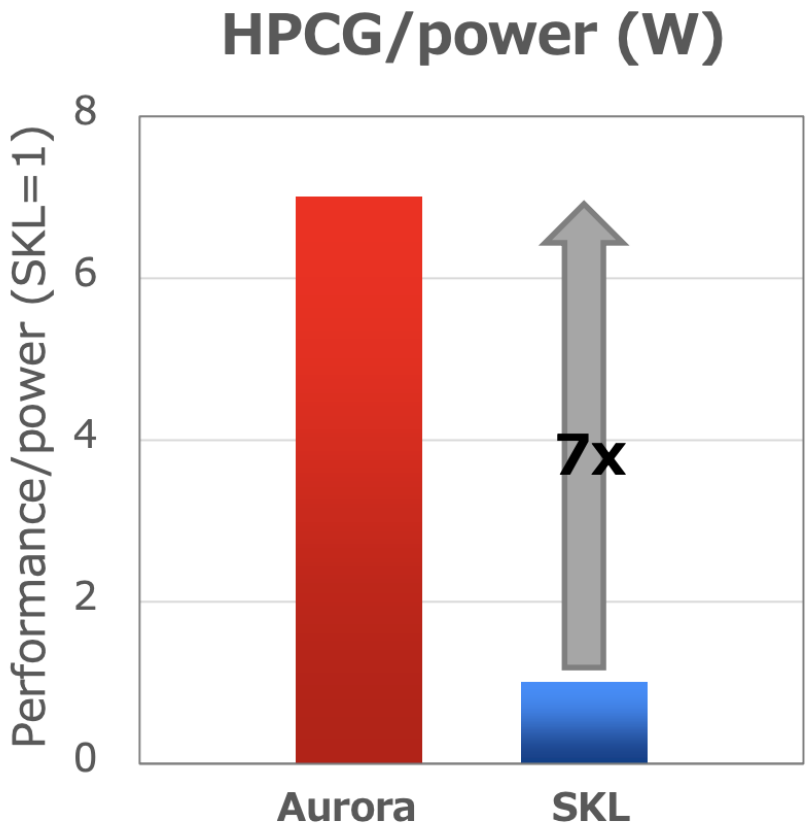


- Aurora is Vector Engine Type 10-B (1.4GHz, 8core)
- SKL is Intel Skylake 6148 Xeon x2/node
- KNL is Intel Knight Landing x1/node
- V100 is NVIDIA Tesla V100 x1/node

Сравнение производительности VE с Xeon Skylake (двухпроцессорная система с Xeon Gold 6148), Intel KNL и Nvidia Tesla V100.

NEC SX-Aurora TSUBASA





- Aurora is Vector Engine Type 10-B (1.4GHz, 8core)
- SKL is Intel Skylake 6148 Xeon x2/node

Сравнение производительности VE с Xeon Skylake (двухпроцессорная система с Xeon Gold 6148).

Векторно-конвейерные компьютеры

Основные особенности архитектуры, дающие заметный вклад в ускорение выполнения программ:

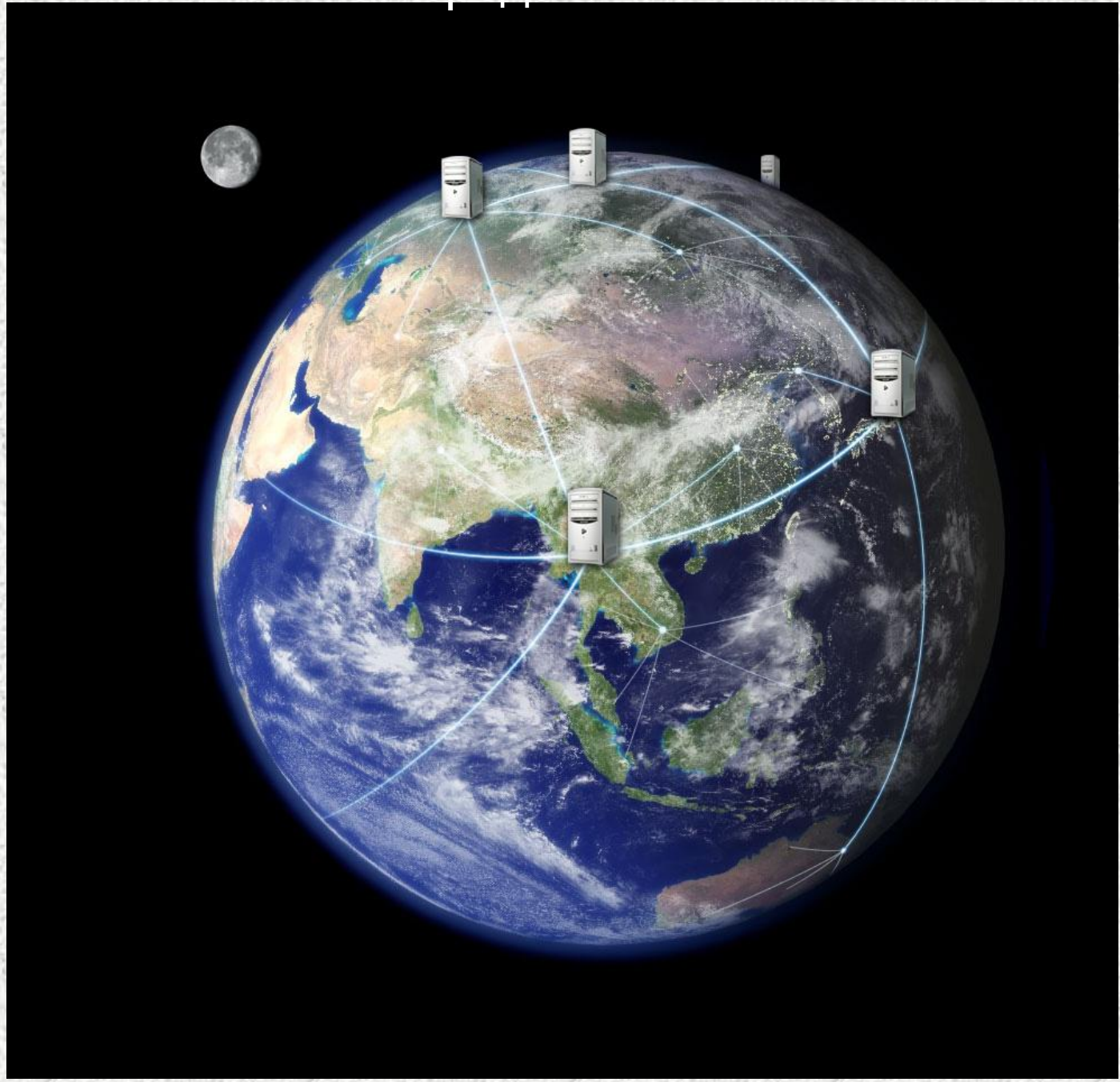
1. *Конвейеризация выполнения команд.*
2. *Независимость функциональных устройств.*
3. *Векторная обработка.*
4. *Дублирование конвейеров векторных устройств и устройств для вещественной арифметики.*
5. *Зацепление функциональных устройств.*
6. *Многопроцессорная обработка.*

Что снижает производительность векторно-конвейерных компьютеров?

1. *Закон Амдала*
2. *Время разгона конвейера*
3. *Секционирование векторных команд*
4. *Конфликты в памяти*
5. *Каналы процессор-память*
6. *Операции чтения/записи в векторные регистры*
7. *Ограниченное число векторных регистров*
8. *Несбалансированное использование устройств*
9. *Отсутствие операции деления*
10. *Перезагрузка буферов команд*
11. *...*

Распределенные вычислительные среды

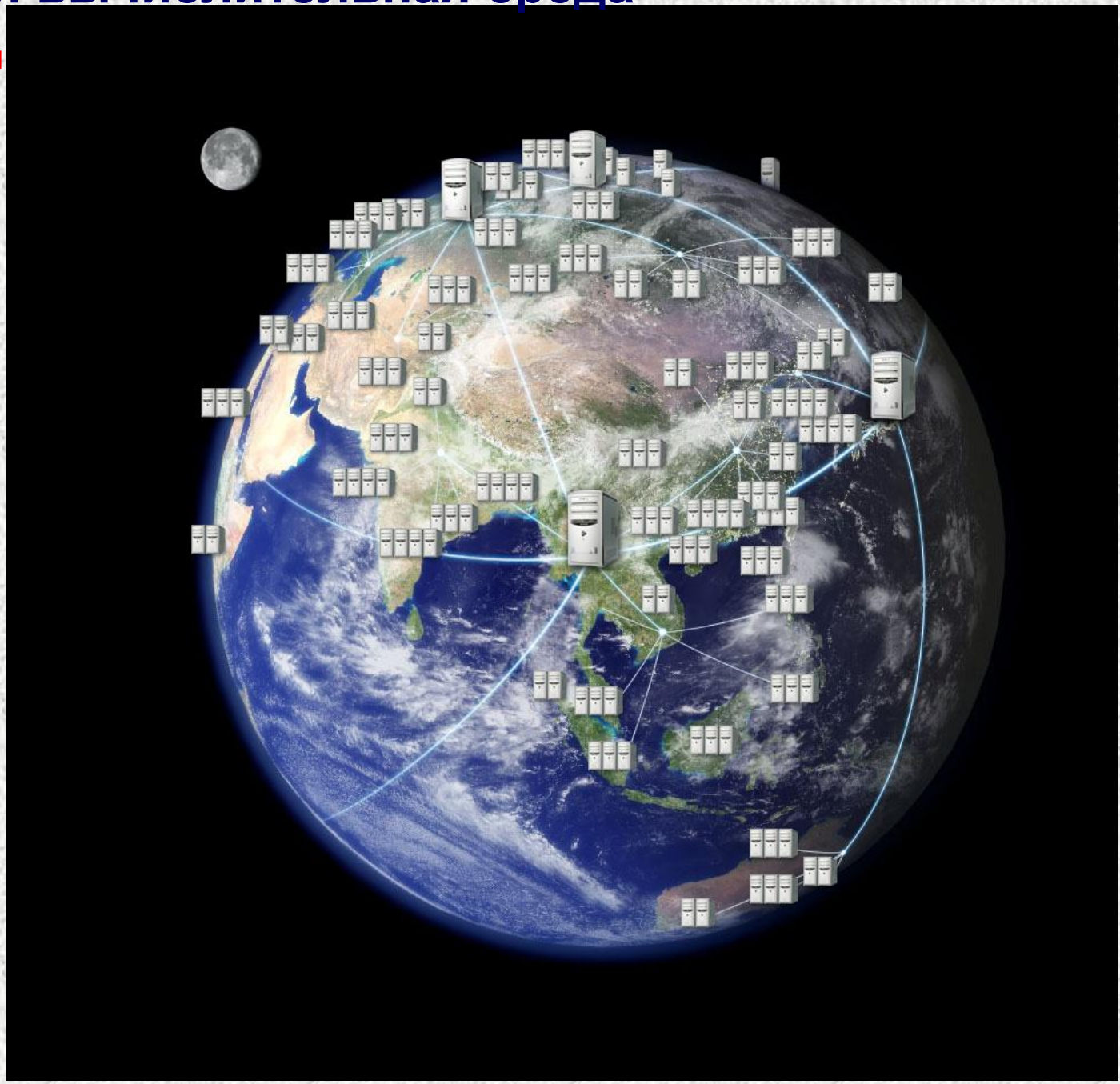
Распределенная вычислительная среда



Распределенные вычислительные среды, в чем отличия от классических компьютеров с распределенной памятью?

Распределенная вычислительная среда

Колоссальные ресурсы

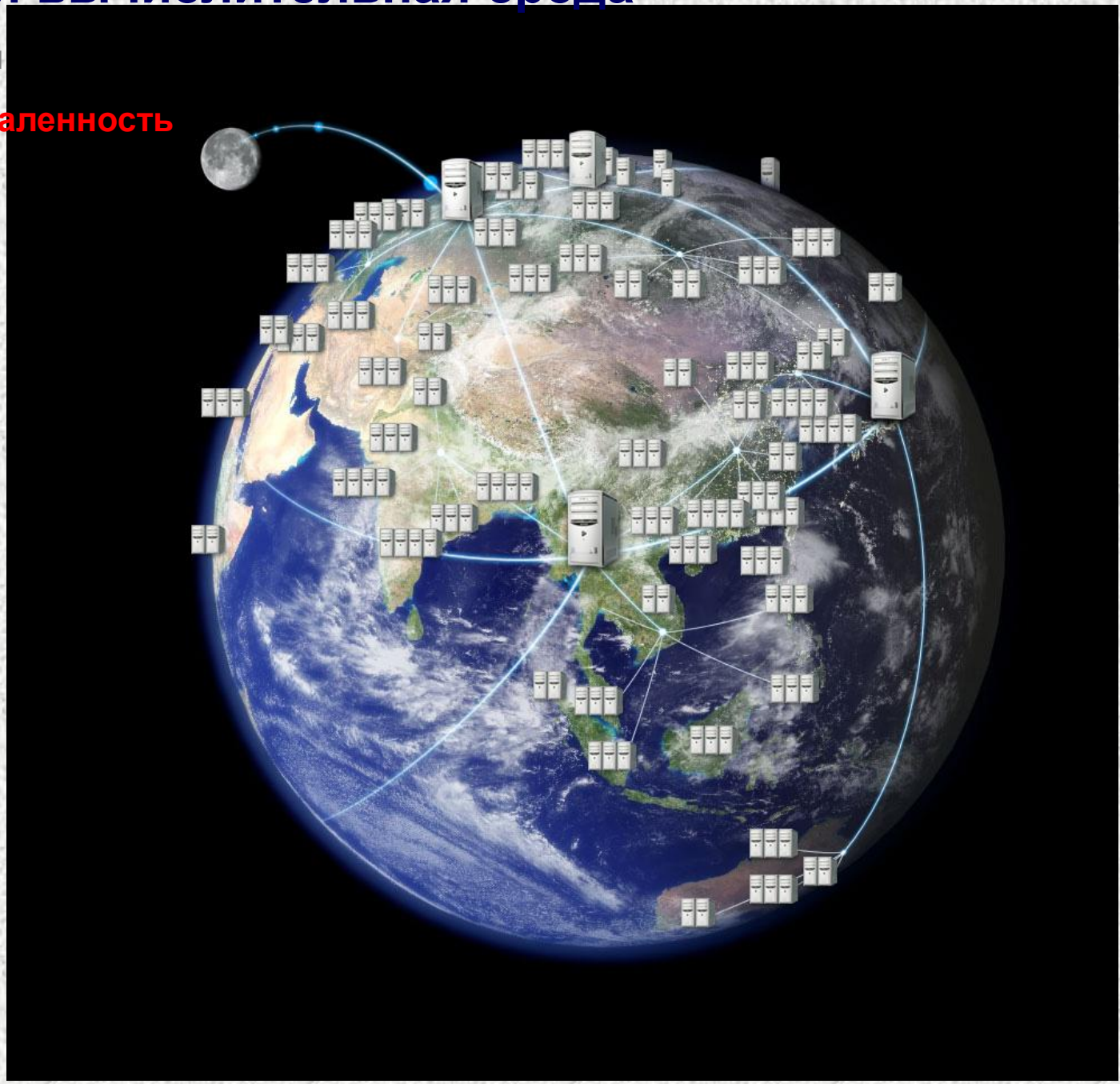


PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

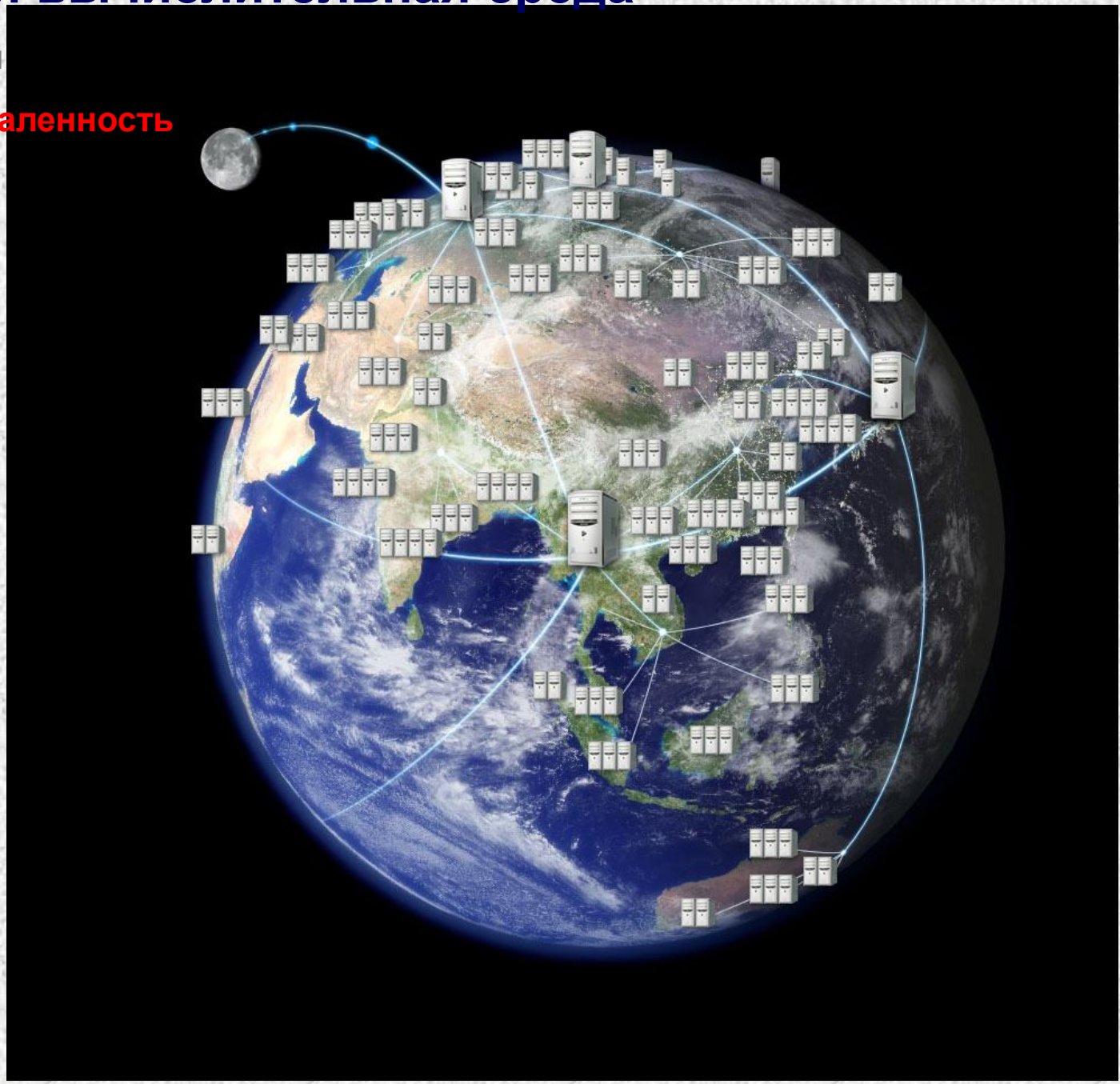


PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

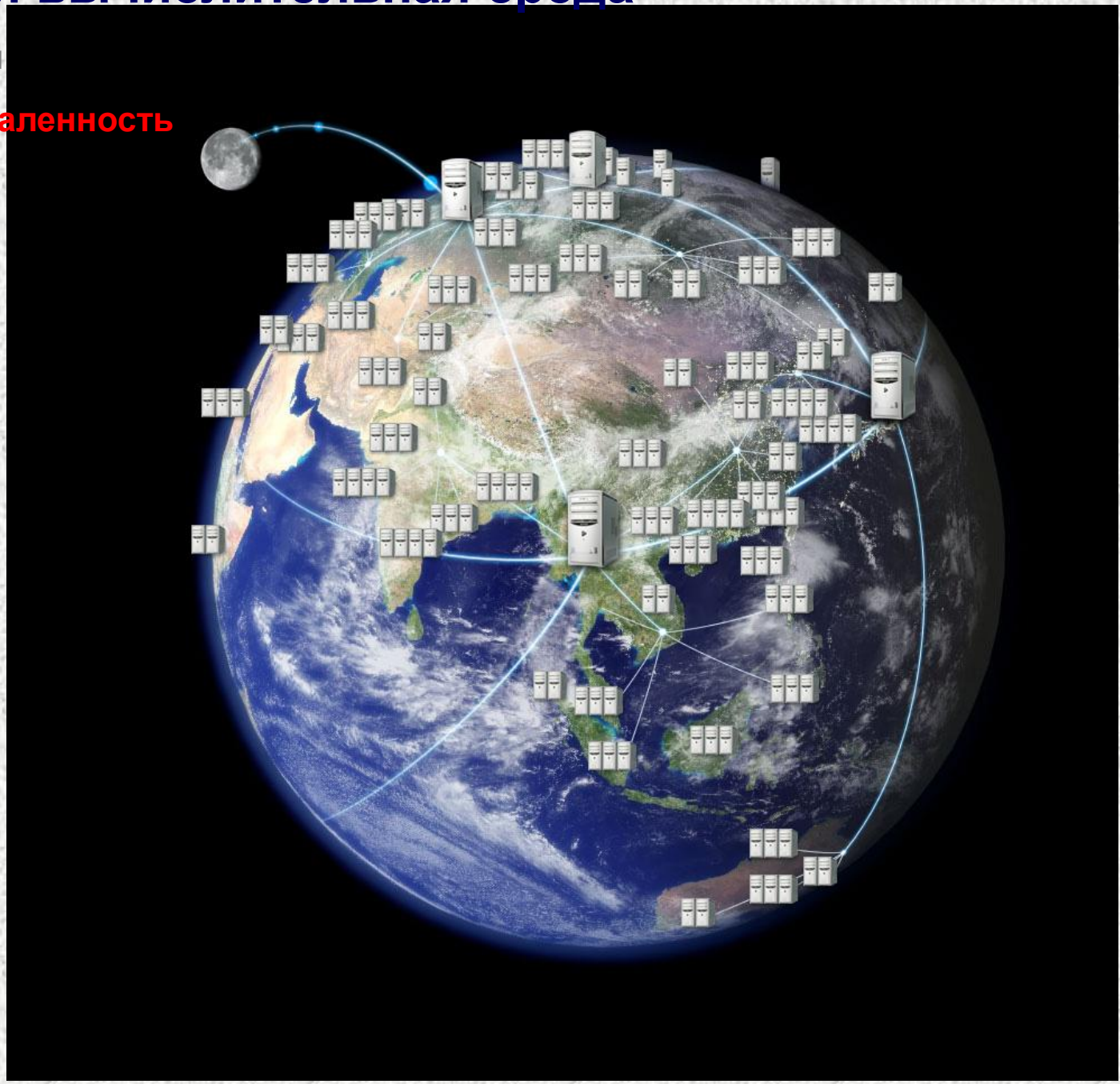


PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

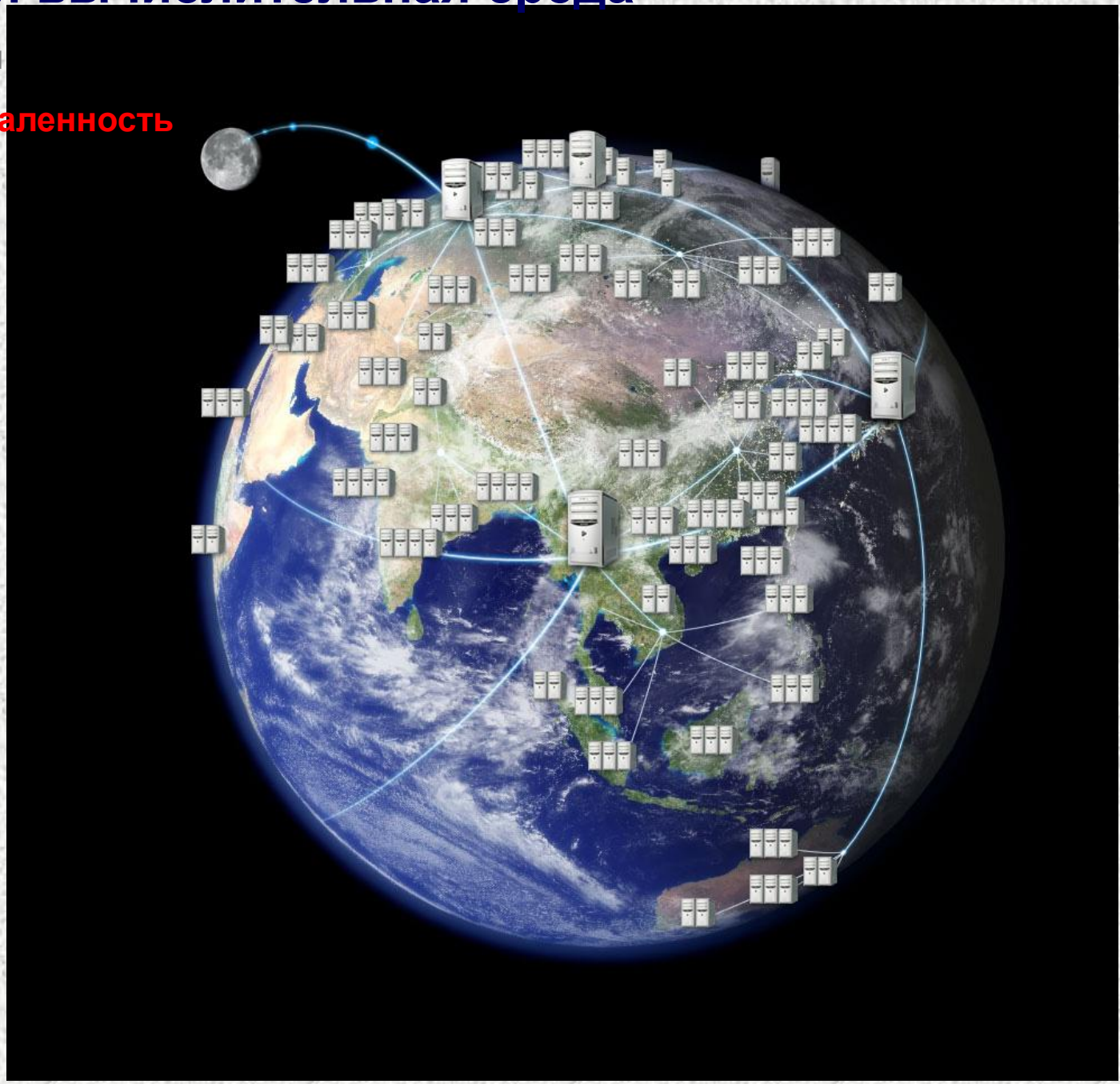


PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность



PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

PARALLEL.RU

Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

PARALLEL.RU

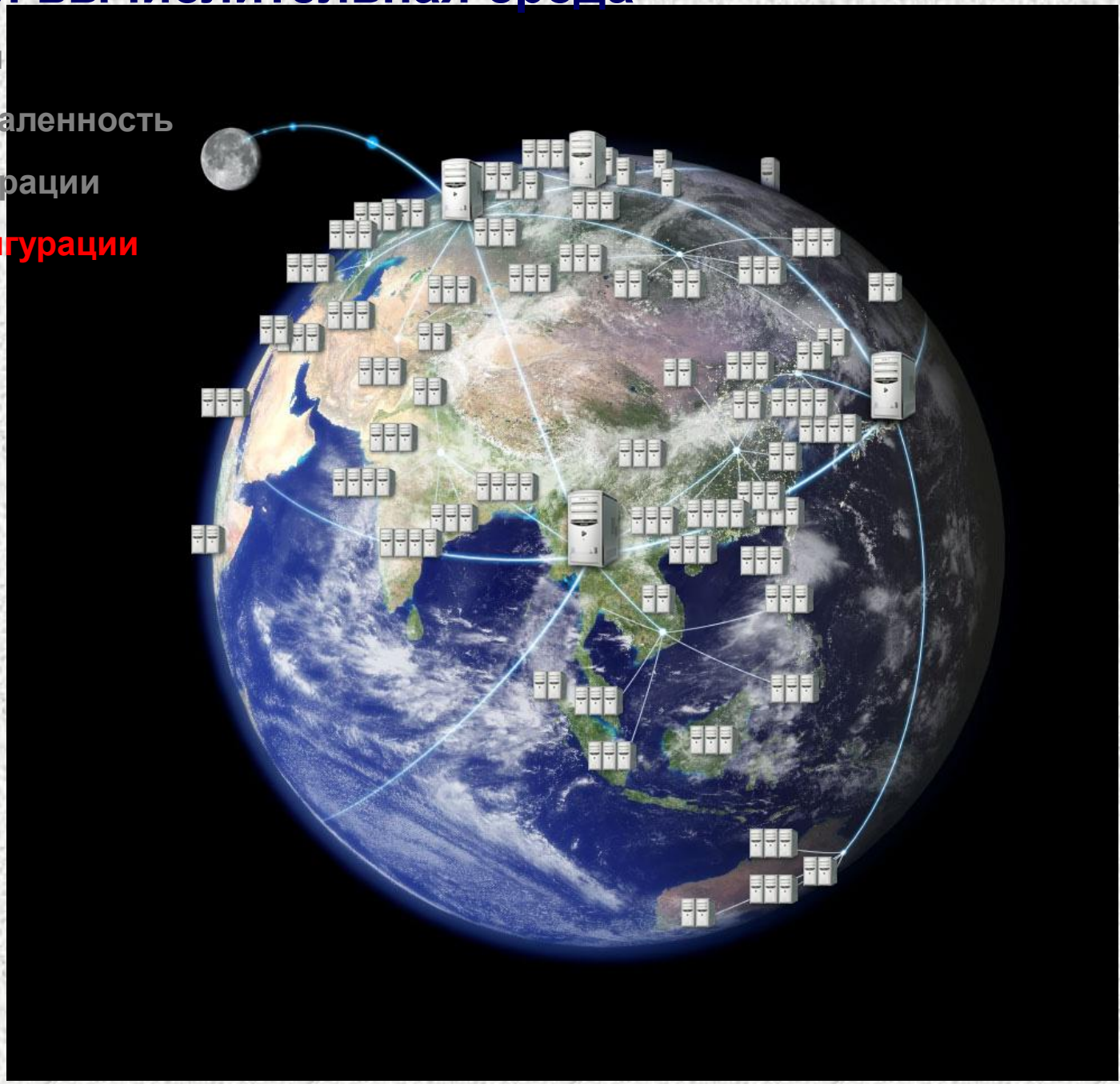
Распределенная вычислительная среда

Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

Неоднородность конфигурации



Распределенная вычислительная среда

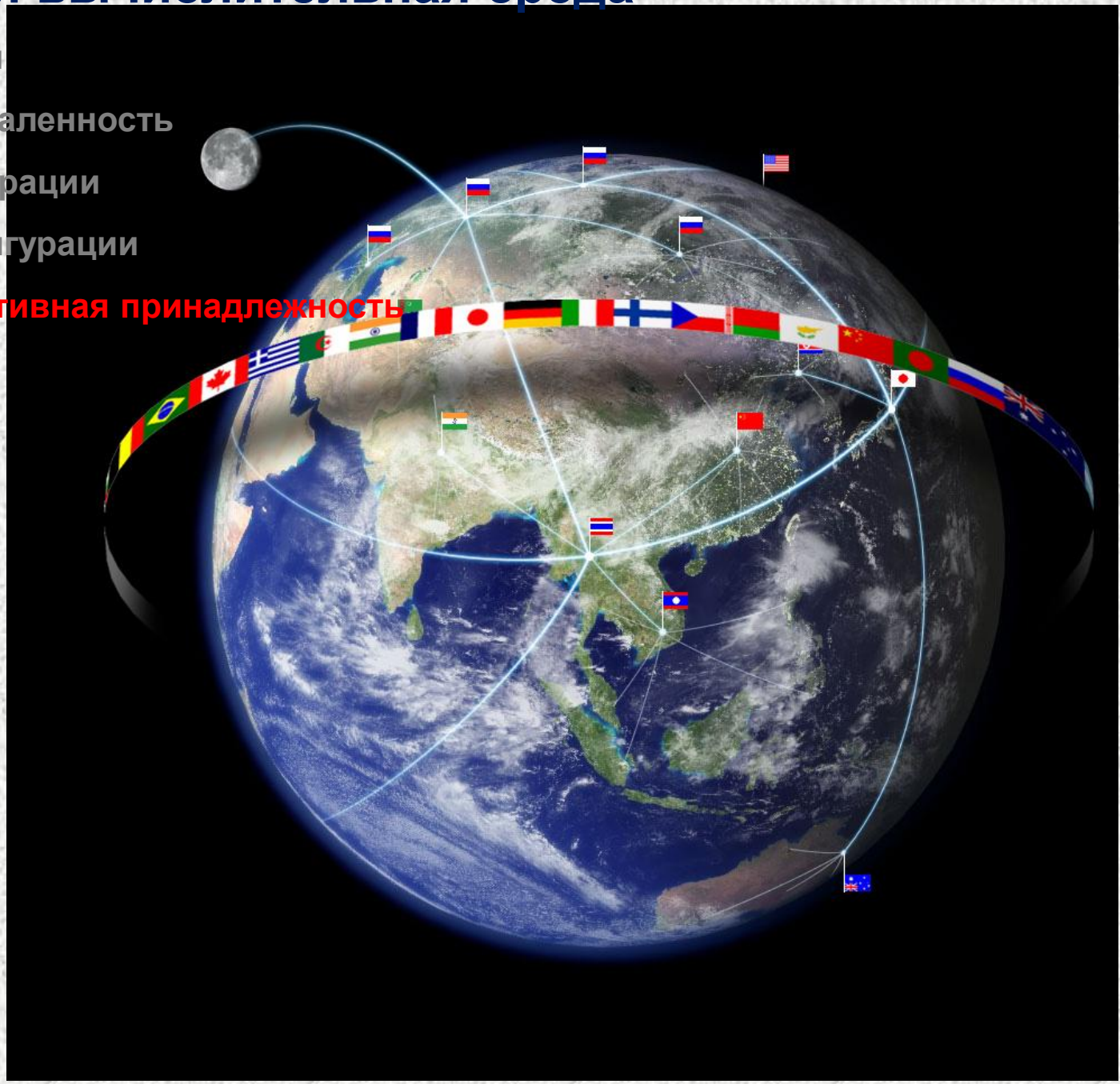
Колоссальные ресурсы

Распределенность и удаленность

Динамичность конфигурации

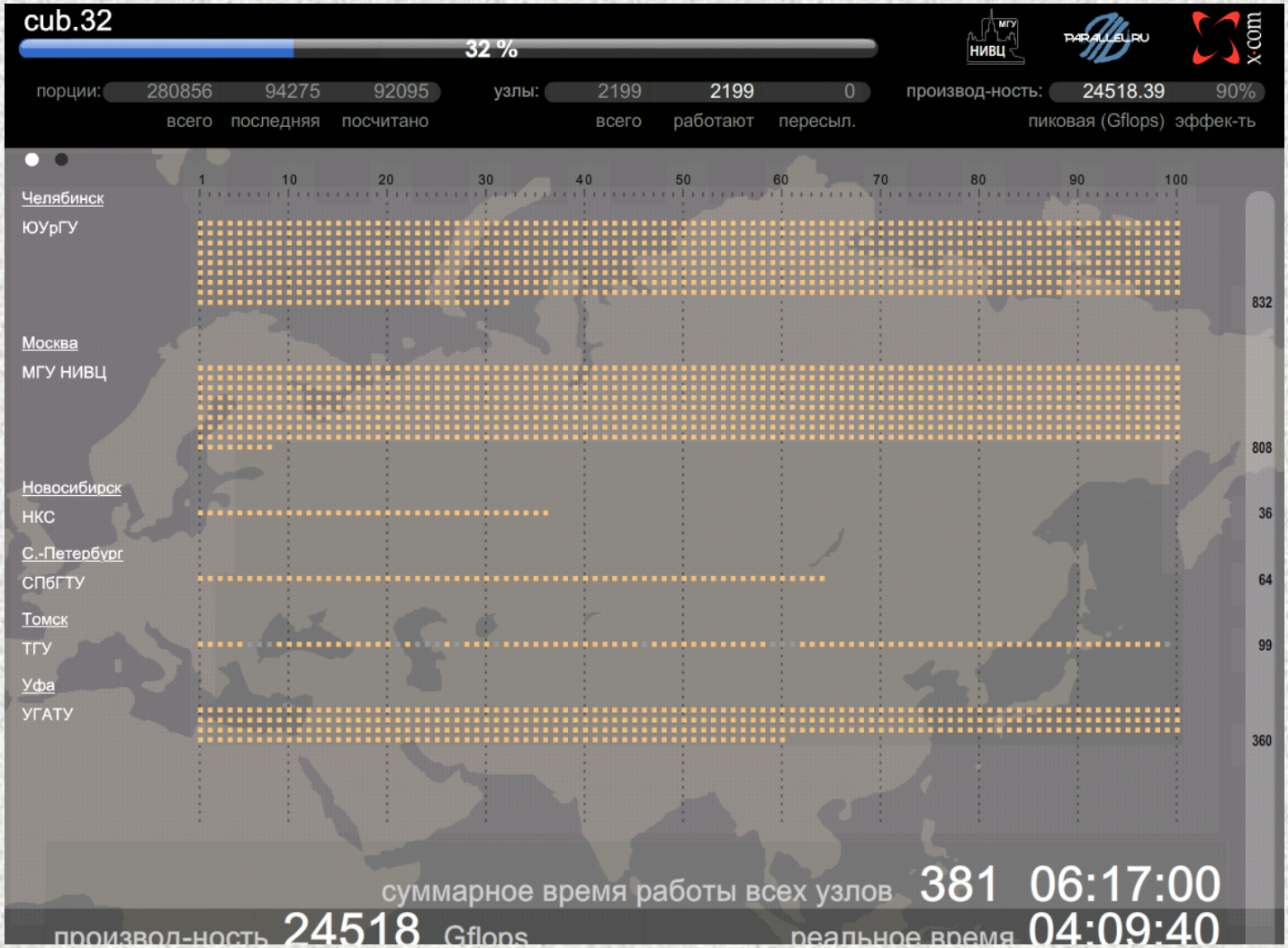
Неоднородность конфигурации

Различная административная принадлежность



Система метакомпьютинга X-COM

(http://x-com.parallel.ru)



x·com

Проектирование новых лекарств

НИВЦ МГУ, Гематологический центр РАМН

Время расчета: 31 дек. 2005 – 11 янв. 2006,

Процессорное время:42774 часа = 1782 дня =

*> **4.5 года** работы одного процессора*

Общее число работавших процессоров: 273,

Максимально процессоров одновременно: 244,

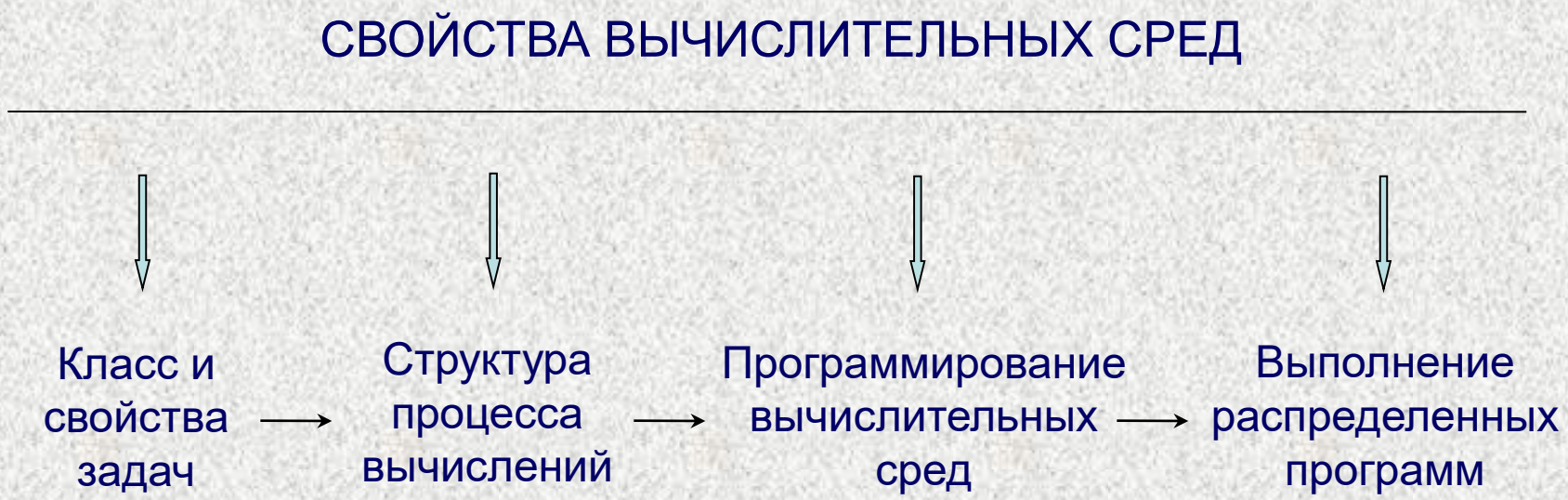
Пиковая производительность: >1 Тфлопс

Вычислительная среда:

кластеры МГУ + кластер ЮУрГУ (г.Челябинск) + учебный класс НИВЦ МГУ + компьютеры в лабораториях

x·com

Использование вычислительных сред



Смежное направление - ***проведение вычислений по требованию (on-demand computing).*** *Для многих организаций экономически невыгодно держать у себя высокопроизводительные компьютеры, поскольку потребности в проведении больших расчетов возникают лишь эпизодически, а стоимость приобретения и сопровождения такого оборудования велика.*

Другой популярный сейчас термин ***облачные вычисления (cloud computing)*** *- это технология распределённой обработки данных, в которой компьютерные ресурсы и мощности предоставляются пользователю как Интернет-сервис. Облачная обработка данных – парадигма, в рамках которой информация постоянно хранится на серверах в Интернет и временно кэшируется на клиентской стороне, например, на персональных компьютерах, игровых приставках, ноутбуках, смартфонах и т.д.*

Многопоточность *(**мультитредовость**, **multithreading**) — свойство платформы или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков, выполняющихся «параллельно», то есть без предписанного порядка во времени. При выполнении некоторых задач такое разделение может достичь более эффективного использования ресурсов вычислительной машины.*

Гипертрейдинг *(**гиперпоточность**, **hyper-threading**) – технология, разработанная компанией Intel для процессоров на микроархитектуре NetBurst. Реализует идею «одновременной мультипоточности». После включения гипертрейдинга один физический процессор (одно физическое ядро) определяется операционной системой как два отдельных процессора (два логических ядра). При определённых рабочих нагрузках позволяет увеличить производительность процессора. Суть технологии: передача полезной работы бездействующим исполнительным устройствам.*

Параллелизм на уровне машинных команд

Параллелизм на уровне машинных команд

Параллелизм на уровне машинных команд *- способность процессора исполнять несколько независимых машинных команд одновременно в рамках одного программного потока (треда).*

Выгоды параллелизма на уровне машинных команд: отсутствие у пользователя необходимости в специальном параллельном программировании; переносимость.

Параллелизм на уровне машинных команд

Суперскалярная архитектура *- архитектура, использующая несколько дешифраторов команд, которые могут нагружать работой множество исполнительных блоков. Если в процессе работы команды, обрабатываемые конвейером, не противоречат друг другу, и одна не зависит от результата другой, то такое устройство может распараллелить выполнение команд.*

Суперскалярность не предполагает, что программа в терминах машинных команд будет включать в себя какую-либо информацию о содержащемся в ней параллелизме. ***Задача обнаружения параллелизма в машинном коде возлагается на аппаратуру,*** *она же и строит соответствующую последовательность исполнения команд. В этом смысле код для суперскалярных процессоров не отражает точно ни природу аппаратного обеспечения, на котором он будет реализован, ни точного временного порядка, в котором будут выполняться команды.*

Параллелизм на уровне машинных команд

VLIW (Very long instruction word, «очень длинное командное слово») *- архитектура процессоров с несколькими вычислительными устройствами. Характеризуется тем, что одна инструкция процессора содержит несколько операций, которые должны выполняться параллельно. Команда VLIW-процессора состоит из набора полей, каждое из которых отвечает за свою операцию. Если какая-то часть процессора на данном этапе не востребована, то соответствующее поле команды не задействуется.*

В процессорах VLIW задача распределения решается во время компиляции и в инструкциях явно указано, какое вычислительное устройство должно выполнять какую команду. ***Компилятор сам выявляет параллелизм в программе*** *и явно сообщает аппаратуре, какие операции не зависят друг от друга. Код для VLIW-процессоров содержит точный план того, как процессор будет выполнять программу: когда будет выполнена каждая операция, какие функциональные устройства будут работать, какие регистры какие операнды будут содержать и т.д.*

Параллелизм на уровне машинных команд

EPIC (Explicitly Parallel Instruction Computing) *- вычисления с явным (заданным) параллелизмом на уровне команд. В этой технологии компилятор явным образом говорит процессору, какие команды можно исполнить параллельно, а какие зависят от других команд.*

В архитектуре EPIC используется концепция длинного командного слова и добавляются следующие черты:

- ***Спекулятивная загрузка данных*** *- вынесение команд загрузки данных далеко вперёд инструкций, использующих эти данные. Это позволяет лучше использовать иерархию памяти. Если произведена запись, затрагивающая загружаемое содержимое, то производится вызов кода восстановления, перезагружающего значение.*
- ***Предикатное выполнение команд*** *- операции выполняются условно, в зависимости от параметра, имеющего булево значение - предиката, связанного с базовым блоком, содержащим операцию. Это позволяет: избежать излишних инструкций ветвления, если количество команд в ветвях условного оператора невелико; уменьшить нагрузку на устройство предсказания переходов.*

Высокопроизводительные компьютерные системы

Какие принципы построения?

Высокопроизводительные компьютерные системы
(процессорная основа)

GP-GPU … *Векторные* *Классические*

FPGAs *Cell* … *Многоядерные*

Эффективность (% от пика),
Производительность на ватт,
Производительность на объем,

…

Простота освоения,
Переносимость программ,
Скорость программирования,


А.В.Каляев
1922 - 2004

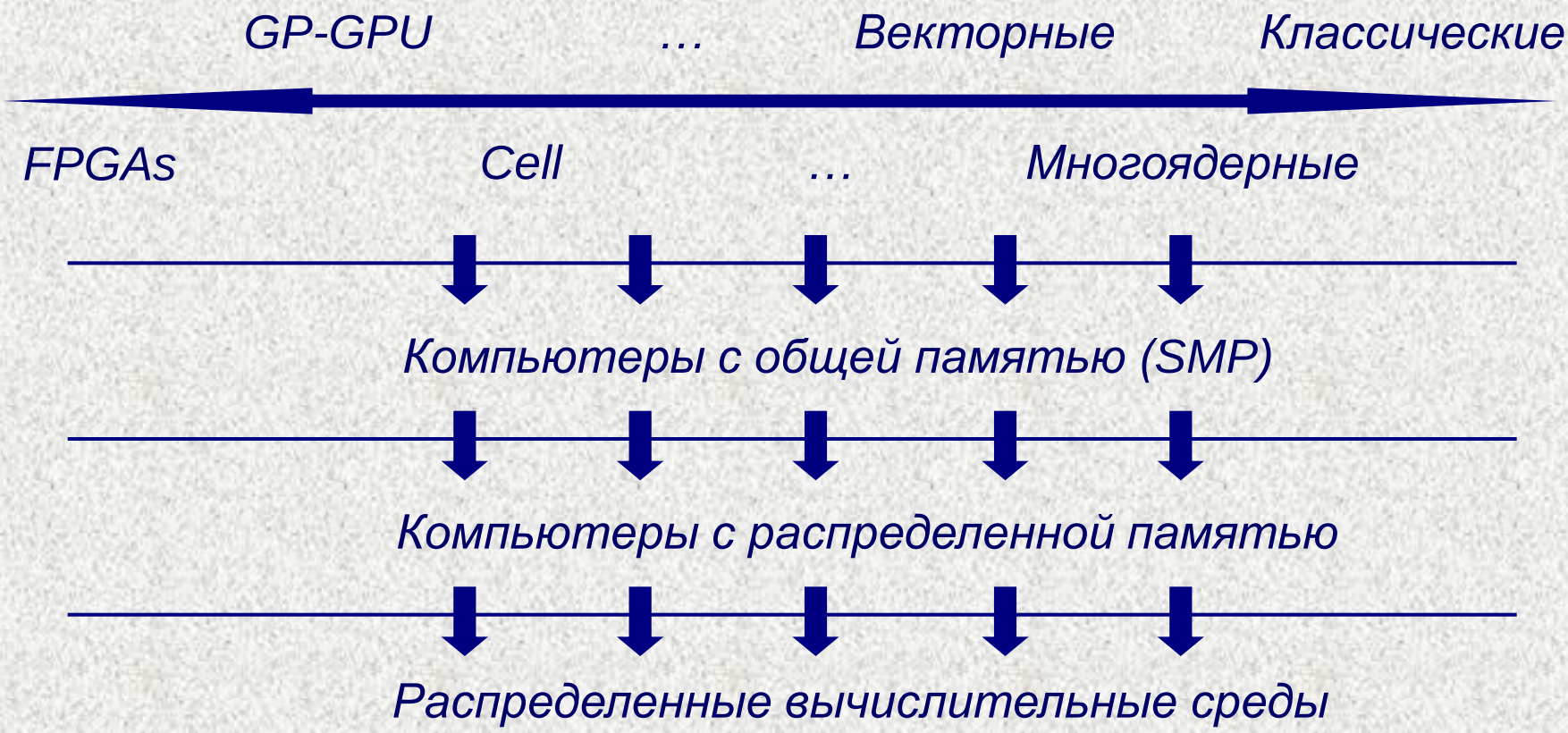

В.С.Бурцев
1927 - 2005


В.А.Мельников
1928 - 1993

Высокопроизводительные компьютерные системы
(процессорная основа)

Процессорная основа компьютеров

Высокопроизводительные компьютерные системы
(процессорная основа)



Московский государственный университет имени М.В.Ломоносова
Факультет Вычислительной математики и кибернетики

СУПЕРКОМПЬЮТЕРЫ И ПАРАЛЛЕЛЬНАЯ ОБРАБОТКА ДАННЫХ (лекция 7)

А.С.Антонов

Вед. н.с. НИВЦ МГУ, к.ф.-м.н.

asa@parallel.ru

ВМК МГУ, 2020