

Суперкомпьютеры и параллельная обработка данных

Бахтин Владимир Александрович
*к.ф.-м.н., ведущий научный сотрудник
Института прикладной математики им М.В.Келдыша
РАН
кафедра системного программирования
факультет вычислительной математики и кибернетики
Московского университета им. М.В. Ломоносова*

Тематический план учебной дисциплины

- Введение в предмет
- Архитектура параллельных вычислительных систем
- Методы оценки производительности параллельных вычислительных систем
- Технологии параллельного программирования
- Введение в теорию анализа структуры программ и алгоритмов
- Введение в параллельные методы решения задач

Литература

- ❑ Лацис А.О. Параллельная обработка данных: учеб. пособие для студ. вузов. Издательский центр «Академия». 2010. Издательство: Академия
- ❑ Якововский М.В. Введение в параллельные методы решения задач. Учебное пособие. Серия: «Суперкомпьютерное образование». Издательство МГУ. 2013.
- ❑ Вл. В. Воеводин, В. В. Воеводин. Параллельные вычисления — СПб., БХВ-Петербург, 2002, 608 с.
- ❑ Антонов А.С. Технологии параллельного программирования MPI и OpenMP: Учеб. пособие. Предисл.: В.А.Садовничий. - Серия «Суперкомпьютерное образование». М.: Издательство Московского университета, 2012.-344 с.

Литература

- ❑ OpenMP Application Programming Interface. Version 5.2. November, 202. URL: <https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5-1.pdf>
- ❑ MPI: A Message-Passing Interface Standard. Version 4.0. June 9, 2021. URL: <https://www.mpi-forum.org/docs/mpi-4.0/mpi40-report.pdf>
- ❑ The OpenACC Application Programming Interface. Version 3.2. November, 2021. URL: <https://www.openacc.org/sites/default/files/inline-images/Specification/OpenACC-3.2-final.pdf>
- ❑ Э. Таненбаум, М. ван Стеен. Распределенные системы. Принципы и парадигмы - М.: Питер, 2003 - 876 с. - Классика computer science; ISBN 5-272-00053-6



PRESENTED BY



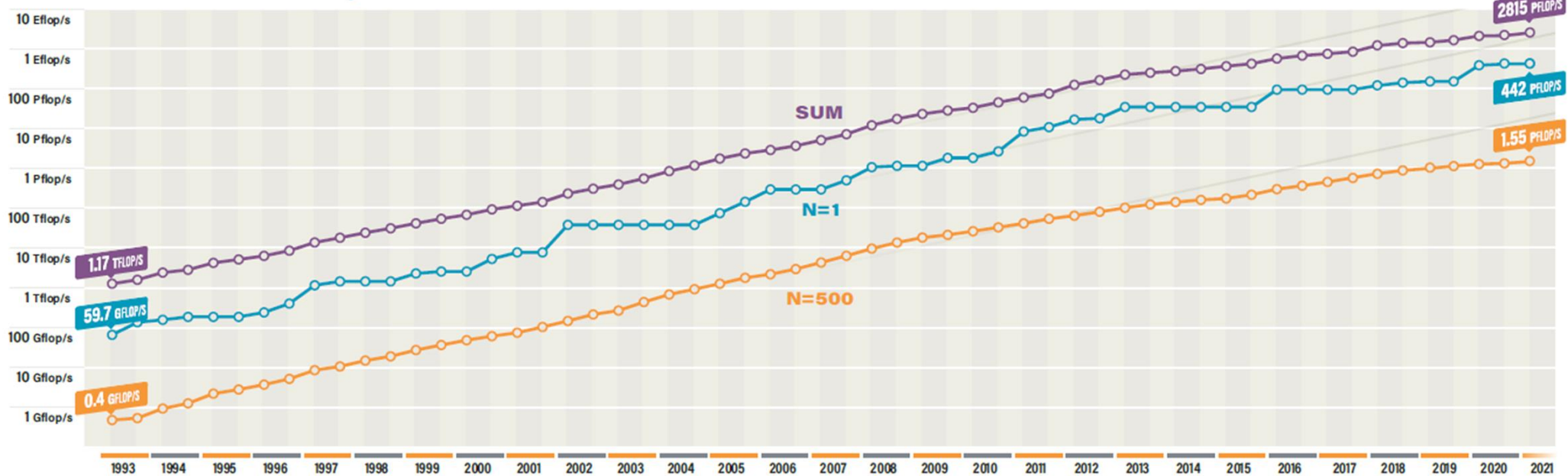
FIND OUT MORE AT

top500.org



JUNE 2021	SYSTEM	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
2	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
3	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44
4	Sunway TaihuLight	Shenwei SW26010 (260C, 1.45 GHz) Custom Interconnect	NSCC in Wuxi	China	10,649,600	93.0	15.4
5	Perlmutter	HPE Cray EX235n, AMD EPYC 7763 (64C, 2.45GHz), NVIDIA A100 SXM4 40 GB, Slingshot-10 (274 GB)	DOE/SC/LBNL	USA	761,856	64.9	2.53

Performance Development



K computer

- ❑ Японский суперкомпьютер производства компании Fujitsu, запущенный в 2011 году в Институте физико-химических исследований в городе Кобе.
- ❑ В июне 2011 года K computer возглавил список самых производительных суперкомпьютеров мира с результатом в тесте LINPACK в 8,162 петафлопс.
- ❑ По состоянию на июнь 2011 года система имела 68 544 8-ядерных процессора SPARC64 VIIIfx, что составляло 548 352 вычислительных ядра, произведенных компанией Fujitsu по 45-нанометровому техпроцессу.
- ❑ В ноябре 2011 года K Computer был достроен, количество процессоров достигло 88 128, а производительность системы на тесте Linpack достигла 10,51 Пфлопс. Таким образом, K Computer стал первым в истории суперкомпьютером, преодолевшим рубеж в 10 Пфлопс.
- ❑ Стоимость 140 миллиардов йен, или 1,2 миллиарда долларов.



Gyokou Supercomputer



<https://youtu.be/-z8ErBIBSo>

8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

7 из 280

Суперкомпьютеры – что это?

- ❑ Суперкомпьютеры – это компьютеры, которые работают значительно быстрее остальной массы современных компьютеров
- ❑ Суперкомпьютеры – это компьютеры, которые занимают большой зал
- ❑ Суперкомпьютеры – это компьютеры, которые весят больше 1 тонны
- ❑ Суперкомпьютеры – это компьютеры, которые стоят больше 1 млн.долл.
- ❑ Суперкомпьютеры – это компьютеры, которые сводят проблему вычислений к проблеме ввода/вывода
- ❑ Суперкомпьютеры – это компьютеры, мощности которых лишь немного не хватает для решения актуальных вычислительно сложных задач

Суперкомпьютеры – что это?

Суперкомпьютером называется вычислительная система, вычислительное быстродействие которой многократно выше, чем у современных ей компьютеров массового выпуска.

Суперкомпьютеры

Серверы

Персональные компьютеры

Мобильные компьютерные устройства



PRESENTED BY



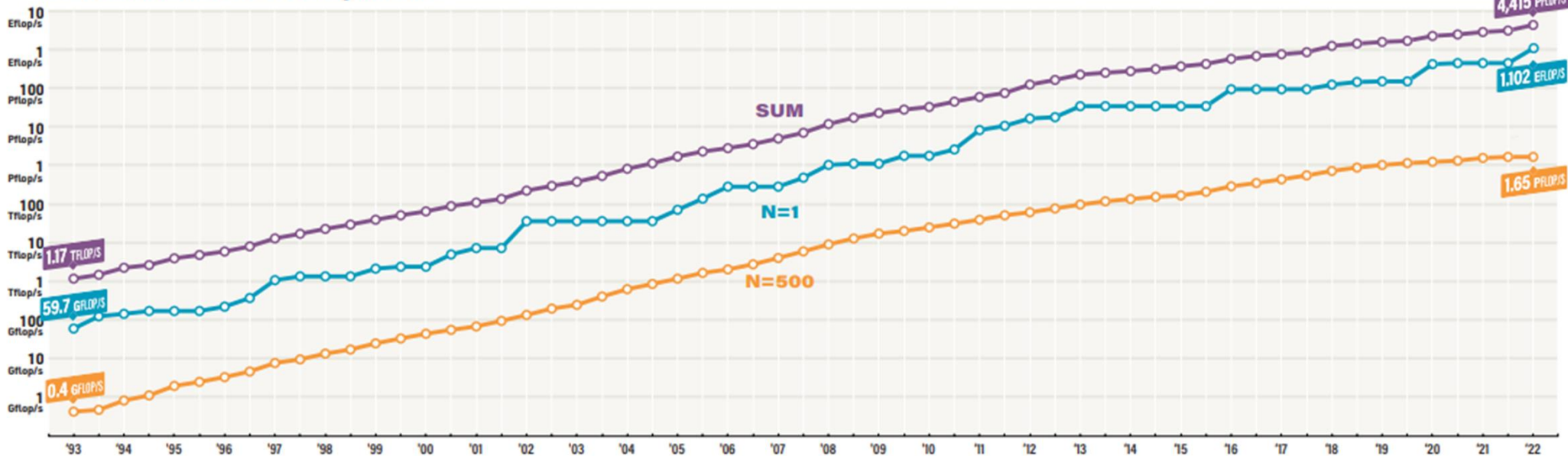
FIND OUT MORE AT

top500.org



JUNE 2022	SYSTEM	SPECS	SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Frontier	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	DOE/SC/ORNL	USA	8,730,112	1,102.0	21.3
2	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
3	LUMI	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	EuroHPC/CSC	Finland	1,268,736	151.9	2.94
4	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
5	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44

Performance Development



8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

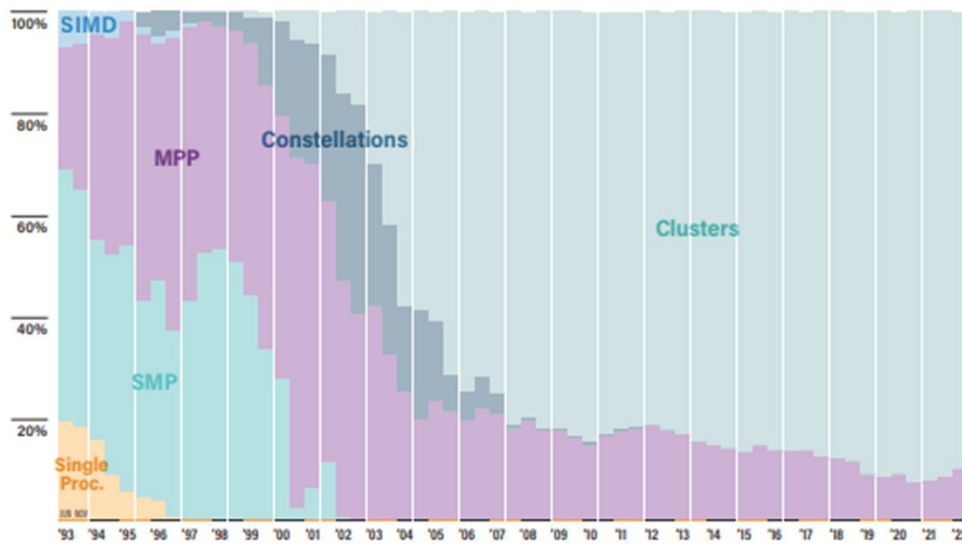
10 из 280

Производительность компьютеров. Тест Linpack

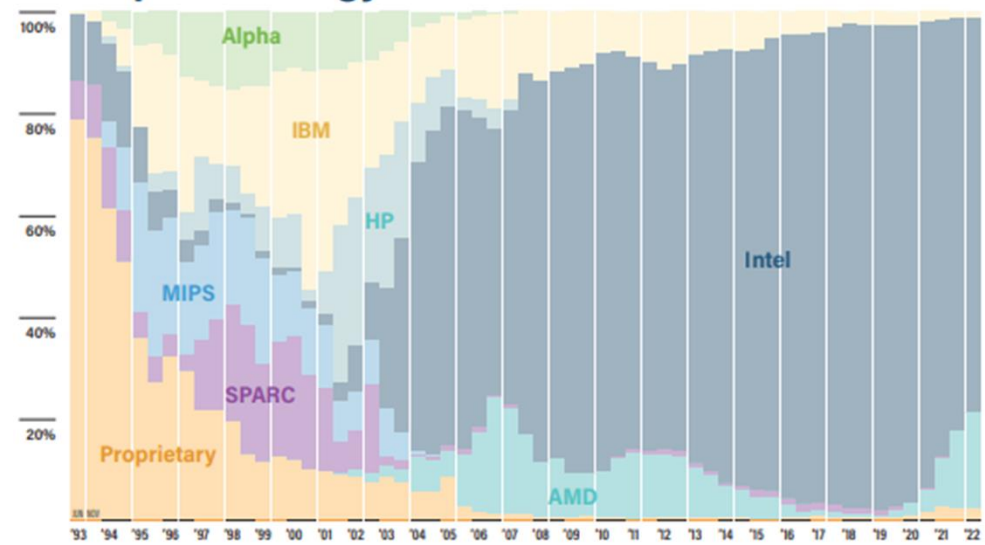
Тест Linpack - решение системы линейных алгебраических уравнений с плотной матрицей.

1. Матрица 100×100 , фиксированный текст программы.
2. Linpack TRP: матрица 1000×1000 , можно менять метод и текст программы. Сложность : $2n^3/3 + 2n^2$.
3. High Performance Linpack: матрица любого размера, множество дополнительных параметров.

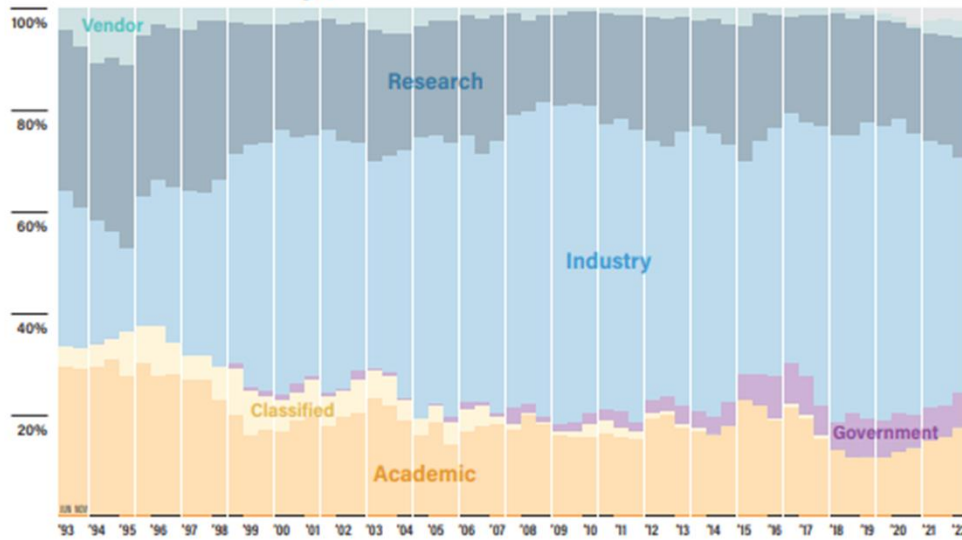
Architectures



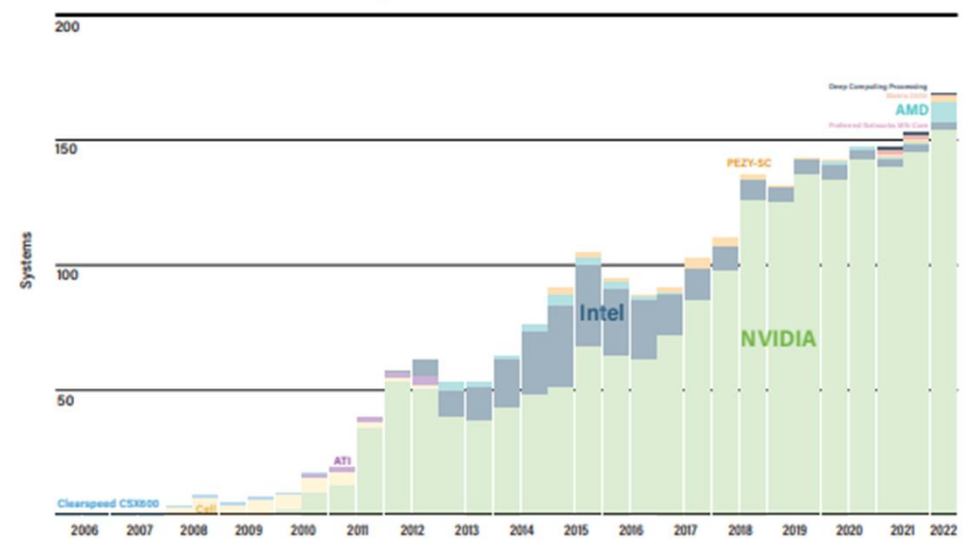
Chip Technology



Installation Type



Accelerators/Co-processors



Поколения архитектур и парадигмы программирования

Середина 70-х годов.

Векторно-конвейерные компьютеры

Особенности архитектуры: векторные функциональные устройства, зацепление функциональных устройств, векторные команды в системе команд, векторные регистры. Программирование: векторизация самых внутренних циклов.

Cray Fortran первый компилятор с Fortran векторизацией

Суперкомпьютер Cray-1
Пиковая производительность
машины — 133 Мфлопса.



Поколения архитектур и парадигмы программирования

Начало 80-х годов.

Векторно-параллельные компьютеры

Особенности архитектуры: векторные функциональные устройства, зацепление функциональных устройств, векторные команды в системе команд, векторные регистры.

Небольшое число процессоров объединяются над общей памятью.

Программирование: векторизация самых внутренних циклов и распараллеливание на внешнем уровне, единое адресное пространство, локальные и глобальные переменные.



Суперкомпьютеры Cray X-MP, Cray Y-MP

Поколения архитектур и парадигмы программирования

Начало 90-х годов.

Массивно-параллельные компьютеры

Особенности архитектуры: тысячи процессоров объединяются с помощью коммуникационной сети по некоторой топологии, распределенная память.

Программирование: обмен сообщениями, отсутствие единого адресного пространства, PVM, Message Passing Interface. Необходимость выделения массового параллелизма, явного распределения данных и согласования параллелизма с распределением.



Суперкомпьютер Cray T3D,
307 Гфлопс

Поколения архитектур и парадигмы программирования

Середина 90-х годов.

Параллельные компьютеры с общей памятью

Особенности архитектуры: сотни процессоров объединяются над общей памятью.

Программирование: единое адресное пространство, локальные и глобальные переменные, OpenMP.



Dec AlphaServer

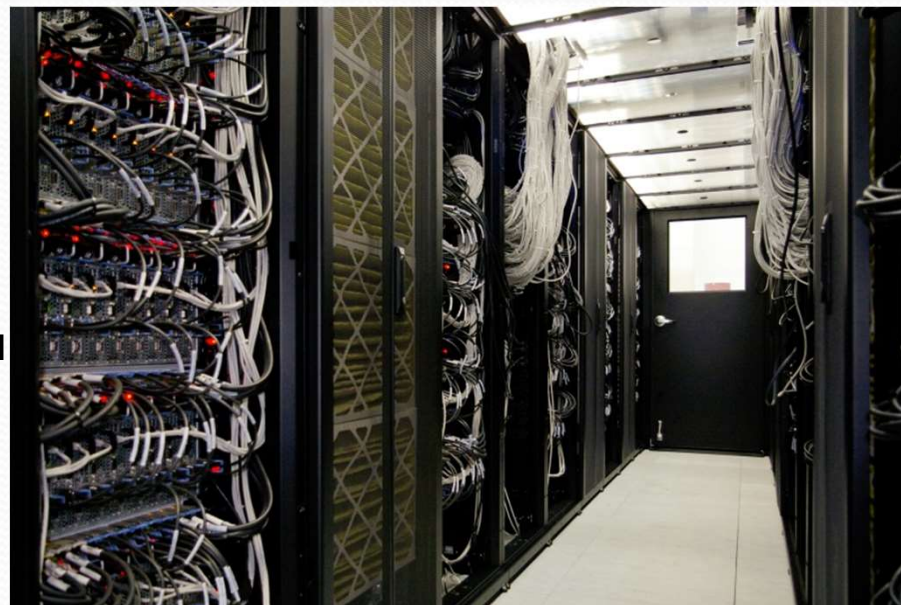
Поколения архитектур и парадигмы программирования

Начало 2000-х.

Кластеры из узлов с общей памятью

Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью.

Программирование: неоднородная схема MPI+OpenMP; необходимость выделения массового параллелизма, явное распределение данных, обмен сообщениями на внешнем уровне; распараллеливание в едином адресном пространстве, локальные и глобальные переменные на уровне узла с общей памятью.



СКИФ МГУ «Чебышев»,
60 Тфлопс

Поколения архитектур и парадигмы программирования

Середина 2000-х.

Кластеры из узлов с общей памятью и ускорителями

Особенности архитектуры: большое число многопроцессорных узлов объединяются вместе с помощью коммуникационной сети по некоторой топологии, распределенная память; в рамках каждого узла несколько (многоядерных) процессоров объединяются над общей памятью; на каждом узле несколько ускорителей (GPU, PHI).

Программирование:

MPI+OpenMP+CUDA/OpenCL



МГУ «Ломоносов», 1.7 Пфлопс

Поколения архитектур и парадигмы программирования

С 1976 года до наших дней:

- 70-е – Векторизация циклов
- 80-е – Распараллеливание циклов (внешних) + Векторизация (внутренних)
- 90-е – MPI
- середина 90-х – OpenMP
- середина 2000-х – MPI+OpenMP
- 2010-е – CUDA, OpenCL, MPI+OpenMP + ускорители (GPU, Xeon Phi)
- ...

50 самых мощных компьютеров СНГ (top50.supercomputers.ru)

№	Название Место установки	Узлов Проц. Ускор.	Архитектура: кол-во узлов: конфигурация узла сеть: вычислительная / сервисная / транспортная	Rmax Rpeak (Тфлп/с)	Разработчик Область применения
1 <i>new</i>	«Червоненкис» Яндекс, Москва	199 398 1592	199: CPU: 2x AMD EPYC 7702 , 1024 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet	21530.0 29415.17	Яндекс NVIDIA IT Services
2 <i>new</i>	«Галушкин» Яндекс, Москва	136 272 1088	136: CPU: 2x AMD EPYC 7702 , 1024 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet	16020.0 20636.1	Яндекс NVIDIA IT Services
3 <i>new</i>	«Ляпунов» Яндекс, Москва	137 274 1096	137: CPU: 2x AMD EPYC 7662, 512 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / нд / 100 Gigabit Ethernet	12810.0 20029.19	NVIDIA Inspur IT Services
4 <i>new</i>	«Кристофари Нео» SberCloud (ООО «Облачные технологии») , СберБанк, Москва	99 198 792	99: CPU: 2x AMD EPYC 7742, 2048 GB RAM Acc: 8x NVIDIA A100 HDR InfiniBand / 10 Gigabit Ethernet / 200 Gigabit Ethernet	11950.0 14908.6	NVIDIA SberCloud (ООО «Облачные технологии») Облачный провайдер
5 ▾	«Кристофари» SberCloud (ООО «Облачные технологии») , СберБанк, Москва	75 150 1200	75: NVIDIA DGX-2 CPU: 2x Intel Xeon Platinum 8168 24C 2.7GHz, 1536 GB RAM Acc: 16x NVIDIA Tesla V100 EDR Infiniband / 100 Gigabit Ethernet / 10 Gigabit Ethernet	6669.0 8789.76	SberCloud (ООО «Облачные технологии») NVIDIA Облачный провайдер
6 ▾	«Ломоносов-2» Московский государственный университет имени М.В.Ломоносова, Москва	1696 1696 1856	1536: CPU: 1x Intel Xeon E5-2697v3, 64 GB RAM Acc: 1x NVIDIA Tesla K40M 160: CPU: 1x Intel Xeon Gold 6126, 96 GB RAM Acc: 2x NVIDIA Tesla P100	2478.0 4946.79	Т-Платформы Наука и образование

8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

20 из 280

Применение суперкомпьютеров

- Сокращение времени решения вычислительно сложных задач
- Сокращение времени обработки больших объемов данных
- Решение задач реального времени
- Создание систем высокой надежности

Суперкомпьютеры... Зачем?

- Неужели есть настолько **сложные задачи**, что для их решения хорошего сервера не хватает?
- Неужели есть настолько **важные задачи**, которые оправдывают крайне высокую стоимость суперкомпьютеров?

А далеко ли вычислительно сложные задачи?

Задача о числе счастливых билетиков :

```

count = 0;
for ( i1 = 0; i1 < 10; ++i1)
  for ( i2 = 0; i2 < 10; ++i2)
    for ( i3 = 0; i3 < 10; ++i3)
      for ( i4 = 0; i4 < 10; ++i4)
        for ( i5 = 0; i5 < 10; ++i5)
          for ( i6 = 0; i6 < 10; ++i6) {
            if ( i1+i2+i3+i4+i5+i6 < 10 )
              count++;
          }

```

Intel Core Duo 2.6 ГГц:
 8 цифр – 0.1 с
 10 цифр – 10 с
 12 цифр – 1780 с



А далеко ли вычислительно сложные задачи?

Задача о числе счастливых билетиков :

```
count = 0;
for ( i1 = 0; i1 < 10; ++i1)
  for ( i2 = 0; i2 < 10; ++i2)
    for ( i3 = 0; i3 < 10; ++i3)
      for ( i4 = 0; i4 < 10; ++i4)
        for ( i5 = 0; i5 < 10; ++i5)
          for ( i6 = 0; i6 < 10; ++i6) {
            if( i1+i2+i3 == i4+i5+i6 )
              count = count+1;
          }
```



*Поможет ли
оптимизация
программы?*

*Поможет ли
использование
суперкомпьютера?*

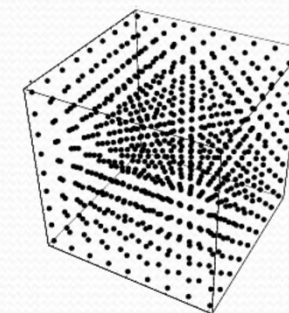
Вычислительная сложность – 10^n операций !

Сверхвысокая производительность - зачем?

Моделирование нефтяных резервуаров:

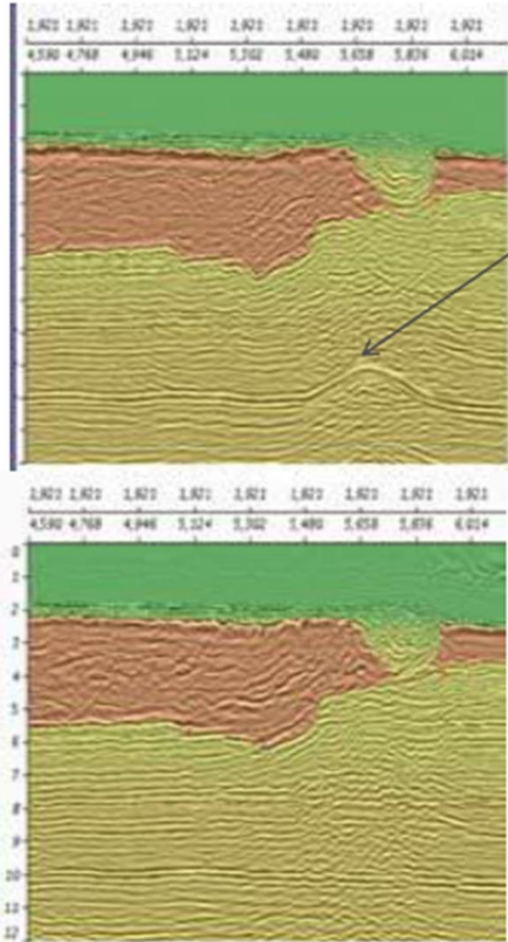
- Нефтеносная область – $100*100*100$ точек
- в каждой точке вычисляется от 5 до 20 функций (скорость, давление, концентрация, температура, ...)
- 200-1000 операций для вычисления каждой функции в каждой точке
- 100-1000 шагов по времени

- Итого: 10^6 (точек сетки) * 10 (функций) *
* 500 (операций) * 500 (шагов) =
= 2500 млрд. операций



Сверхвысокая производительность - зачем?

Ex: Increasing efficiency in Oil & Gas



Seismic profiles of a region of the Gulf of Mexico.

The top image, in 2003 , on 64 processors,
At the bottom right-hand side , a structure shaped like a bowler hat,
typical of a petroleum zone.

Based on this image, ready to install boring equipment on this site.

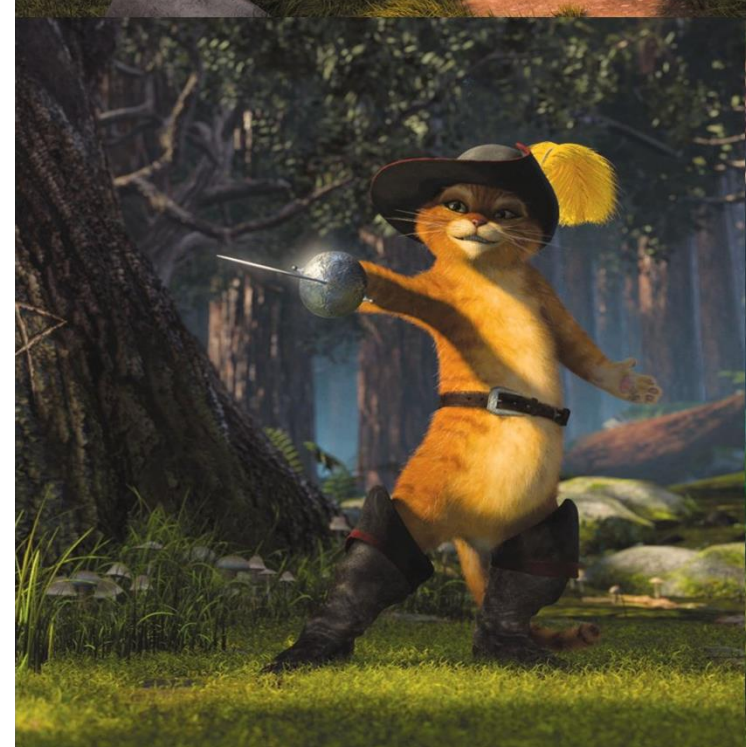
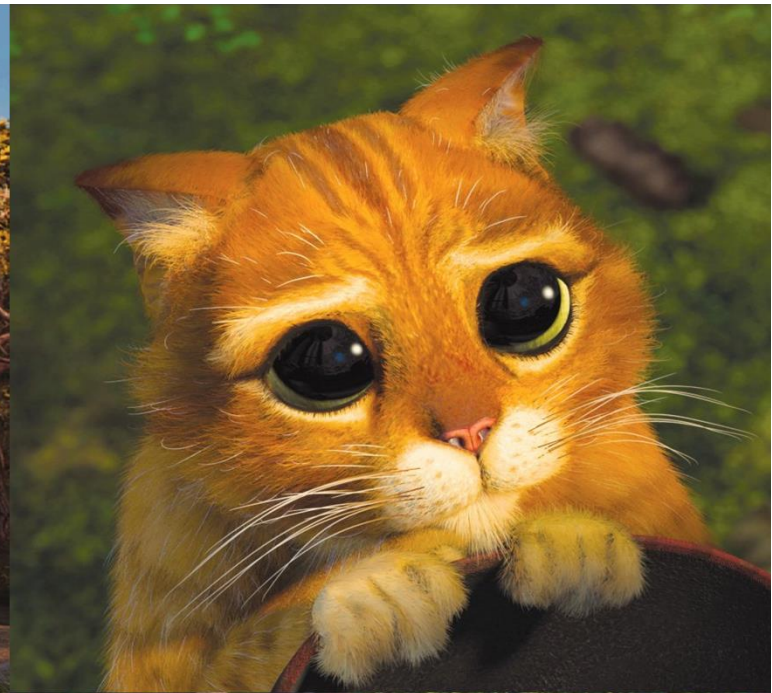
Fresh data analysis, on a 13 000 cores supercomputer revealed
that the structure was an artefact.

Thanks to HPC, 80 M\$ saved

In the **mid-90's**, **only 40%** of deposits fulfilled their promises.
Numerical simulations that analyse data obtained by seismic
echography have radically changed the playing field. Armed with
the new supercomputer ... , the Total engineers are **now** hitting
the bull's eye **in 60-70% of cases."**

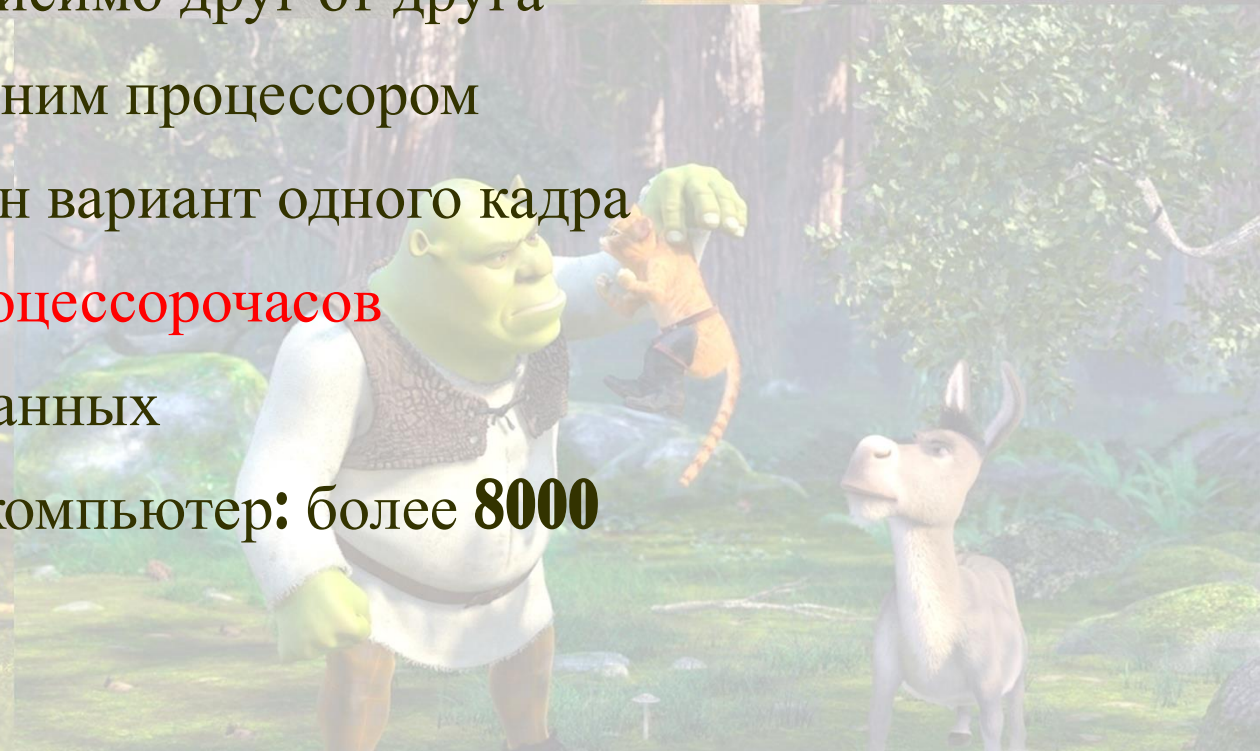
Journal La Recherche, special HPC, July 2009,

Acknowledgements: H. Calandra, Ph. Ricoux (TOTAL)



ШРЕК ТРЕТИЙ. Суперкомпьютерный...

- **24** кадра в секунду
- более **120 000** кадров в фильме
- обработка кадров независимо друг от друга
- кадр обрабатывается одним процессором
- в среднем **2** часа на один вариант одного кадра
- всего: более **20 млн. процессорочасов**
- всего: более **30** Тбайт данных
- использованный суперкомпьютер: более **8000** процессорных ядер



Сверхвысокая производительность - зачем?

ruDALL-E - сеть для создания изображения на основе текстового описания на русском языке (<https://rudalle.ru/>)

«шахматная ладья из изумрудного материала»



«кошка, одетая в корону»



Сверхвысокая производительность - зачем?

ruDALL-E - сеть для создания изображения на основе текстового описания на русском языке (<https://rudalle.ru/>)

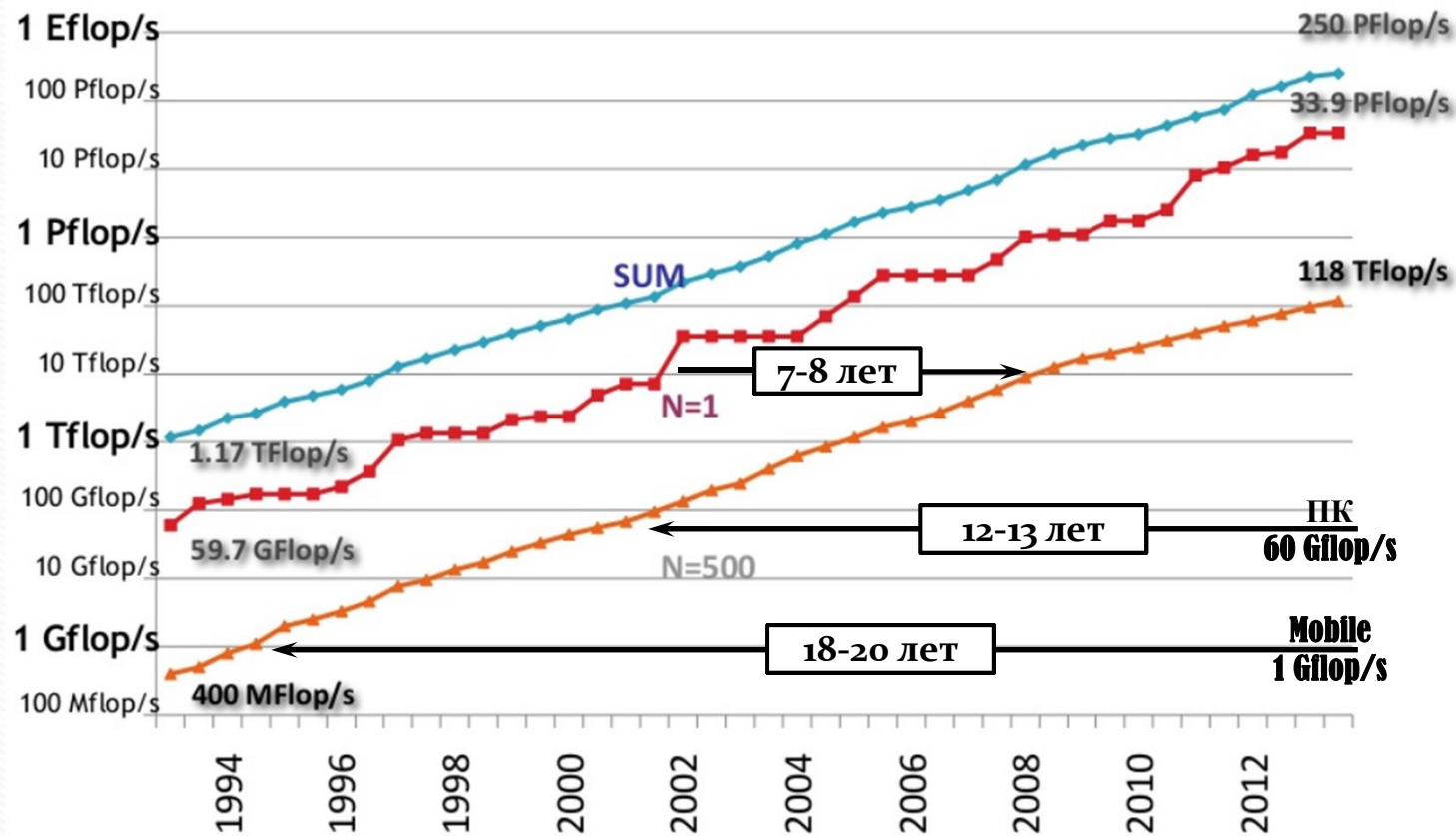
Создание изображений происходит в три этапа: сначала одна нейросеть принимает текст на вход и генерирует заданное число картинок, затем следующая выбирает наиболее удачные из них и соответствующие описанию, а третья увеличивает их в размере без потери качества.

Два варианта модели:

- ruDALL-E XL, которая содержит 1,3 миллиарда параметров;
- ruDALL-E 12B с 12 миллиардами параметров.

Обучение заняло 23 тысячи GPU-часов на массиве данных из 120 миллионов пар «текст-изображение»

Рост производительности



<http://linpack.hpc.msu.ru/>

Важные сокращения

Мега (Mega) – 10^6 (миллион)

Гига (Giga) – 10^9 (биллион / миллиард)

Тера (Tera) – 10^{12} (триллион)

Пета (Peta) – 10^{15} (квадриллион)

Экса (Exa) – 10^{18} (квинтиллион)

*Флоп/с, Flop/s – Floating point operations
per second*

*15 Tflop/s = $15 * 10^{12}$ арифметических операций
в секунду над вещественными данными,
представленными в форме с плавающей точкой.*

Важные сокращения

Мега (Mega) – 10^6 (миллион)

Гига (Giga) – 10^9 (биллион / миллиард)

Тера (Tera) – 10^{12} (триллион)

Пета (Peta) – 10^{15} (квадриллион)

Экса (Exa) – 10^{18} (квинтиллион)

Флоп/с, Flop/s – *Floating point operations per second*

15 Tflop/s = $15 * 10^{12}$ **арифметических** операций в секунду над **вещественными** данными, представленными в форме с **плавающей точкой**.

Годы, флопсы и степень параллелизма (когда и как был достигнут очередной 'X'flops)

10^6 Mflops	1964 Г.	CDC 6600	10 MHz	1 CPUs
10^9 Gflops	1985 Г.	Cray 2	125 MHz	8 CPUs
10^{12} Tflops	1997 Г.	ASCI Red	200 MHz	9152 CPUs
10^{15} Pflops	2008 Г.	Roadrunner	3,2 GHz	122400 Cores
10^{18} Eflops	2022 Г.	Frontier	2 GHz CPUs (606,208 cores) and 37,888 GPUs (8,335,360 cores)	

Увеличение производительности компьютеров: за счет чего?

*EDSAC, 1949 год
год*

Cray Titan, 2012

изменение

такт: $2 \cdot 10^{-6}$ с

$\approx 4.4 \cdot 10^3$

$4.5 \cdot 10^{-10}$ с (2.2 GHz)

произв.: 10^2 оп/с

$\approx 1.7 \cdot 10^{14}$

$1.7 \cdot 10^{16}$ оп/с

Время такта = $1/(\text{тактовая частота})$

Увеличение производительности компьютеров: за счет чего?

*EDSAC, 1949 год
год*

Cray Titan, 2012

изменение

такт: $2 \cdot 10^{-6}$ с

$\approx 4.4 \cdot 10^3$

$4.5 \cdot 10^{-10}$ с (2.2 GHz)

произв.: 10^2 оп/с

$\approx 1.7 \cdot 10^{14}$

$1.7 \cdot 10^{16}$ оп/с

Два вывода.

1. Безусловно, без развития элементной базы не было бы такого прогресса в развитии компьютеров.

2. Но основной вклад в увеличении производительности компьютеров – это развитие архитектуры, и прежде всего, за счет глубокого внедрения идей параллелизма

Тенденции развития современных процессоров

В течение нескольких десятилетий развитие ЭВМ сопровождалось удвоением их быстродействия каждые 1.5-2 года. Это обеспечивалось и повышением тактовой частоты и совершенствованием архитектуры (параллельное и конвейерное выполнение команд).

Узким местом стала оперативная память. Знаменитый закон Мура, так хорошо работающий для процессоров, совершенно не применим для памяти, где скорости доступа удваиваются в лучшем случае каждые 5-6 лет.

Совершенствовались системы кэш-памяти, увеличивался объем, усложнялись алгоритмы ее использования.

Для процессора Intel Itanium:

Latency to L1: 1-2 cycles

Latency to L2: 5 - 7 cycles

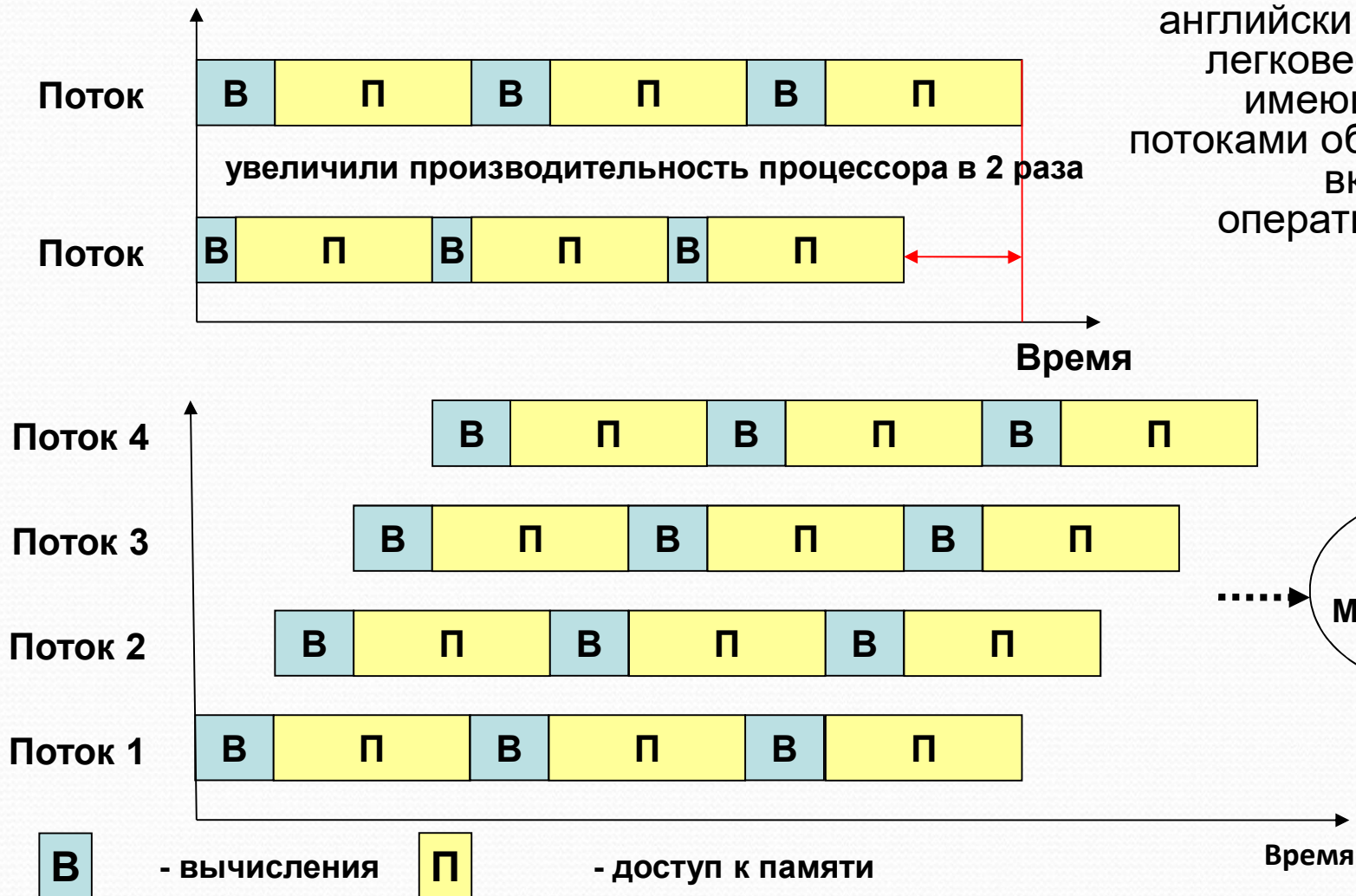
Latency to L3: 12 - 21 cycles

Latency to memory: 180 – 225 cycles

Важным параметром становится - **GUPS** (Giga Updates Per Second)

Тенденции развития современных процессоров

Поток или **нить** (по-английски “thread”) – это легковесный процесс, имеющий с другими потоками общие ресурсы, включая общую оперативную память.





PRESENTED BY



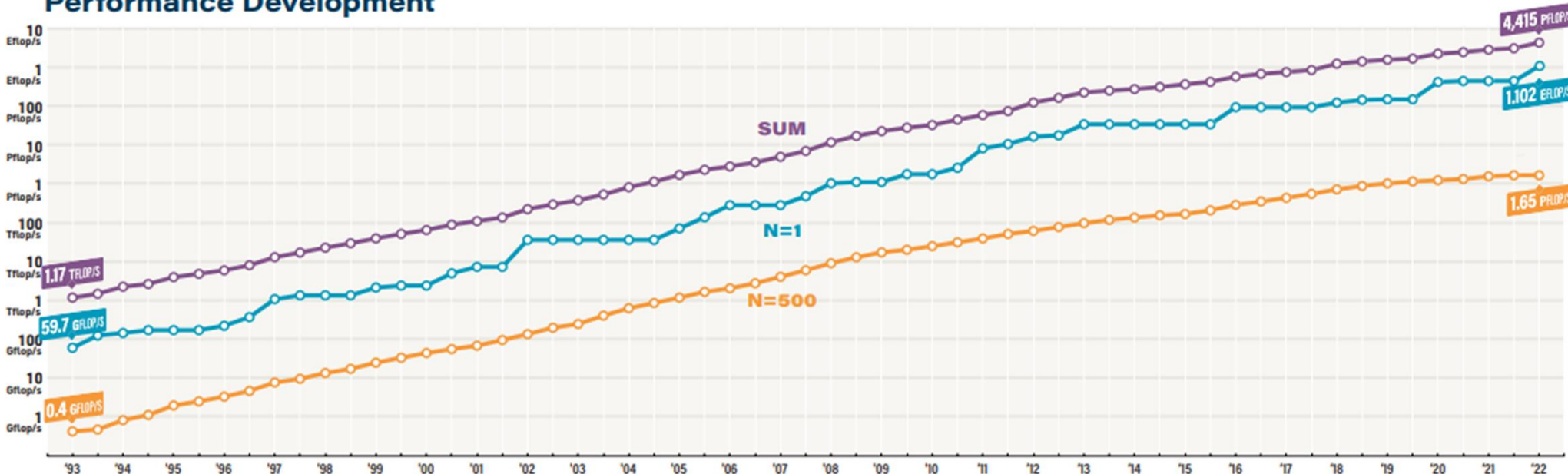
FIND OUT MORE AT

top500.org



JUNE 2022	SYSTEM	SPECS	SITE	COUNTRY	CORES	RMAX PFLOP/S	POWER MW
1	Frontier	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	DOE/SC/ORNL	USA	8,730,112	1,102.0	21.3
2	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
3	LUMI	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	EuroHPC/CSC	Finland	1,268,736	151.9	2.94
4	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
5	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44

Performance Development



8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

39 из 280

Суперкомпьютерные системы (Top500)

№ 32 в Top 500

**Суперкомпьютер MARU, ThinkSystem SD650 V2, Xeon Platinum 8368Q
38C 2.6GHz, Infiniband HDR**

- ❑ Пиковая производительность – 25495,14 TFlop/s
- ❑ Число ядер в системе — 306 432
- ❑ Производительность на Linpack - 16753 TFlop/s (65.71 % от пиковой)
- ❑ Энергопотребление комплекса - **15414 кВт**

Суперкомпьютерные системы (Top500)

№ 6 в Top 500

Суперкомпьютер Sunway TaihuLight, Sunway MPP, SW26010 260C
1.45GHz, Custom interconnect

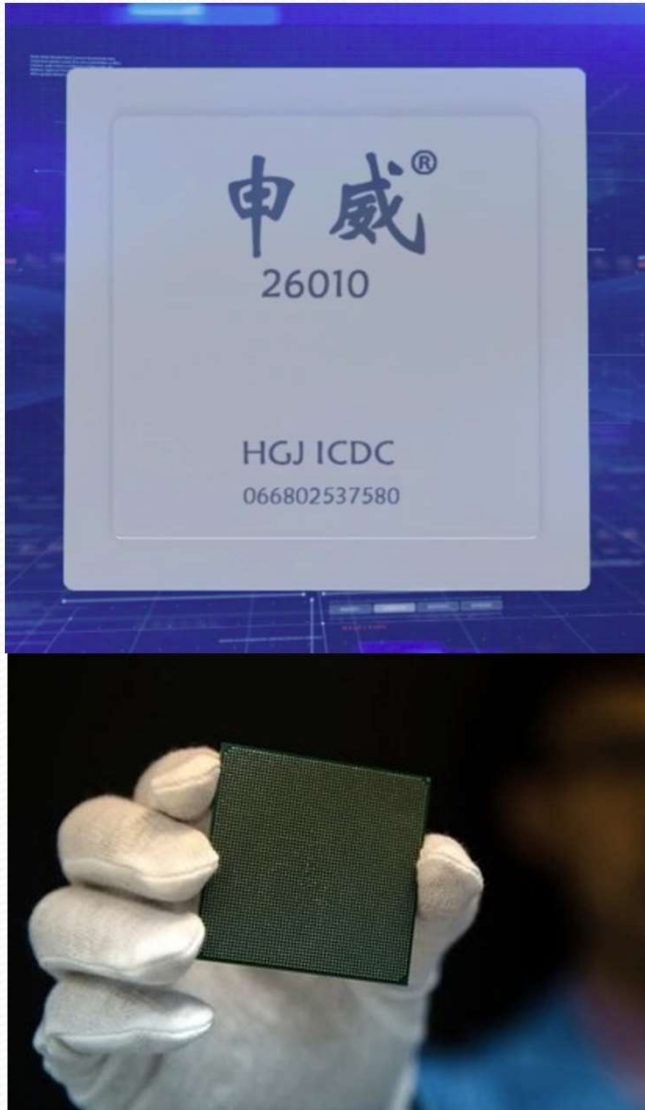
- ❑ Пиковая производительность – 125435.9 TFlop/s
- ❑ Число ядер в системе — 10 649 600
- ❑ Производительность на Linpack – 93014.5 TFlop/s (74.15 % от пиковой)
- ❑ Энергопотребление комплекса - **15371 кВт**

Важным параметром становится – **Power Efficiency (GFlops/watt)**

6,05 VS 1,09

Как добиться максимальной производительности на Ватт => Chip
MultiProcessing, многоядерность.

Тенденции развития современных процессоров



ShenWei SW26010

64-разрядный RISC-процессор с поддержкой SIMD-инструкций и внеочередным исполнением команд

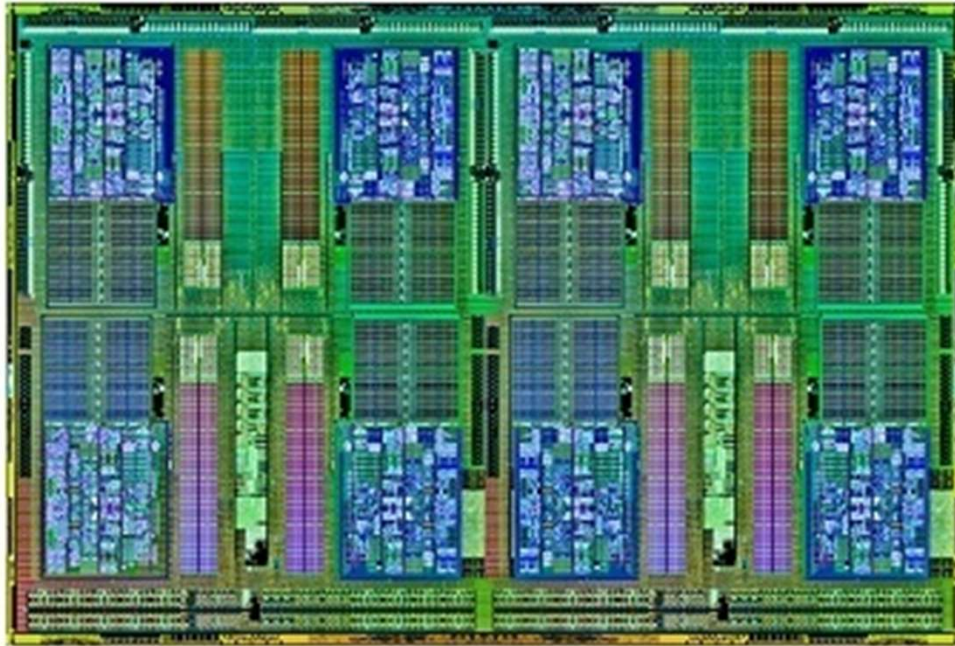
Изготовлен по схеме, предусматривающей использование четырех кластеров с 64 вычислительными ядрами (CPE) и одним управляющим ядром (MPE) в каждом.

В каждом кластере также имеется собственный контроллер памяти, суммарная пропускная способность на один процессорный разъем достигает 136,5 ГБ/с.

На каждое ядро выделено 12 КБ кэш-памяти инструкций и 64 КБ кэш-памяти данных.

Рабочая частота процессора - 1,45 ГГц.

Тенденции развития современных процессоров



AMD Opteron серии 6300

6380 SE 16 ядер @ 2,5 ГГц, 16 МБ L3 Cache

6348 12 ядер @ 2,8 ГГц, 16 МБ L3 Cache

6328 8 ядер @ 3,2 ГГц, 16 МБ L3 Cache

6308 4 ядра @ 3,5 ГГц, 16 МБ L3 Cache

технология AMD Turbo CORE

встроенный контроллер памяти (4 канала памяти DDR3)

4 канала «точка-точка» с использованием HyperTransport 3.0

AMD EPYC 7003 Series Processors

AMD EPYC™ 7763

of CPU Cores 64
of Threads 128
Max Boost Clock Up to 3.5GHz
Base Clock 2.45GHz
Default TDP / TDP 280W

AMD EPYC™ 75F3

of CPU Cores 32
of Threads 64
Max Boost Clock Up to 4.0GHz
Base Clock 2.95GHz
Default TDP / TDP 280W

AMD EPYC™ 7713

of CPU Cores 64
of Threads 128
Max Boost Clock Up to 3.6GHz
Base Clock 2.0GHz
Default TDP / TDP 225W

AMD EPYC™ 7643

of CPU Cores 48
of Threads 96
Max Boost Clock Up to 3.6GHz
Base Clock 2.3GHz
Default TDP / TDP 225W

AMD EPYC™ 7543

of CPU Cores 32
of Threads 64
Max Boost Clock Up to 3.7GHz
Base Clock 2.8GHz
Default TDP / TDP 225W

<https://www.amd.com/en/processors/epyc-7003-series>

Процессоры AMD EPYC серии Milan-X

Процессор	Ядер/Потоков	Базовая частота	Турбо	TDP	Кэш L3 (L3 + 3D V-Cache)
EPYC 7773X	64/128	2,2 ГГц	3,5 ГГц	280 Вт	768 МБ
EPYC 7573X	32/64	2,8 ГГц	3,6 ГГц	280 Вт	768 МБ
EPYC 7473X	24/48	2,8 ГГц	3,7 ГГц	240 Вт	768 МБ
EPYC 7373X	16/32	3,05 ГГц	3,8 ГГц	240 Вт	768 МБ



8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

45 из 280

Тенденции развития современных процессоров

Intel Xeon Processor серии E5

E5-2699 v4 (55M Cache, 2.20 GHz) 22 ядра, 44 нити

E5-2698 v4 (50M Cache, 2.20 GHz) 20 ядер, 40 нитей

E5-2697 v4 (45M Cache, 2.30 GHz) 18 ядер, 36 нитей

E5-2697A v4 (40M Cache, 2.60 GHz) 16 ядер, 32 нити

E5-2667 v4 (25M Cache, 3.20 GHz) 8 ядер, 16 нитей

Intel® Turbo Boost

Intel® Hyper-Threading

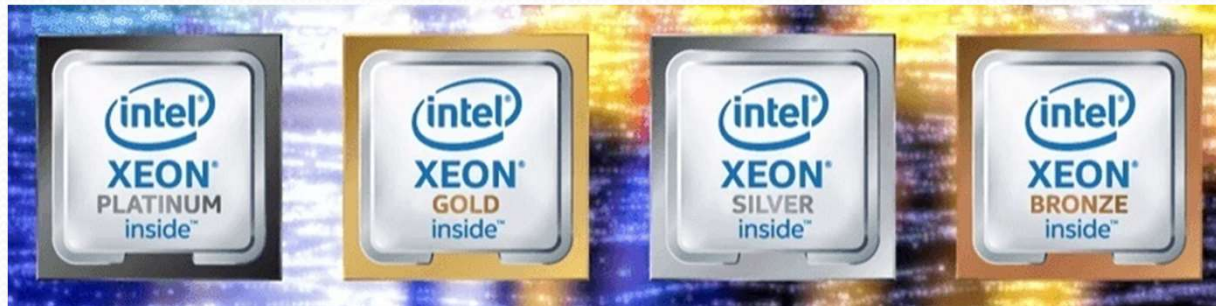
Intel® Intelligent Power

Intel® QuickPath



3rd Generation Intel Xeon Scalable Processors

Processor	Launch Date	# of cores	Max Turbo Frequency	Processor Base Frequency	Cache	TDP
Intel Xeon Platinum 8368	Q2'21	38	3.40 GHz	2.40 GHz	57 MB	270 W
Intel Xeon Platinum 8368Q	Q2'21	38	3.70 GHz	2.60 GHz	57 MB	270 W
Intel Xeon Platinum 8380	Q2'21	40	3.40 GHz	2.30 GHz	60 MB	270 W
Intel Xeon Platinum 8360Y	Q2'21	36	3.50 GHz	2.40 GHz	54 MB	250 W
Intel Xeon Platinum 8358	Q2'21	32	3.40 GHz	2.60 GHz	48 MB	250 W
Intel Xeon Platinum 8380H	Q2'20	28	4.30 GHz	2.90 GHz	38.5 MB	250 W

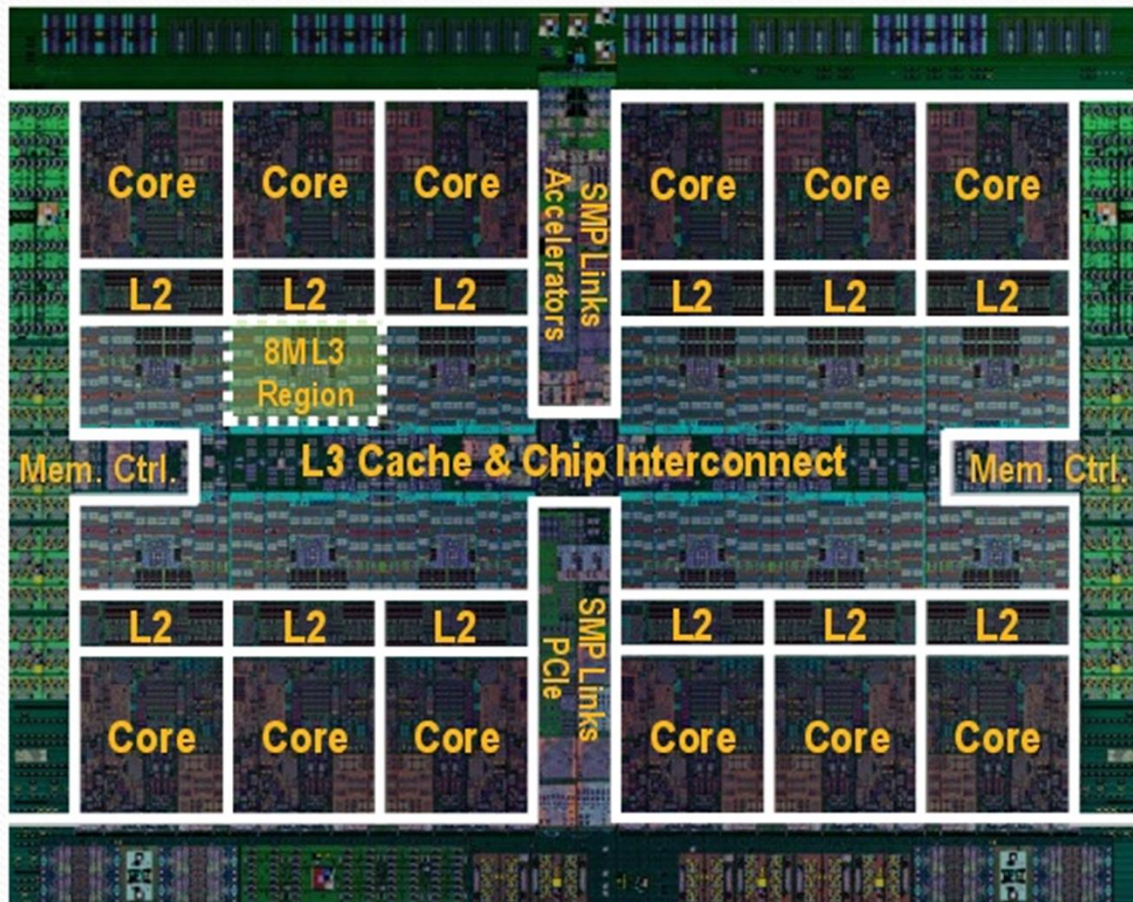


8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

47 из 280

Тенденции развития современных процессоров

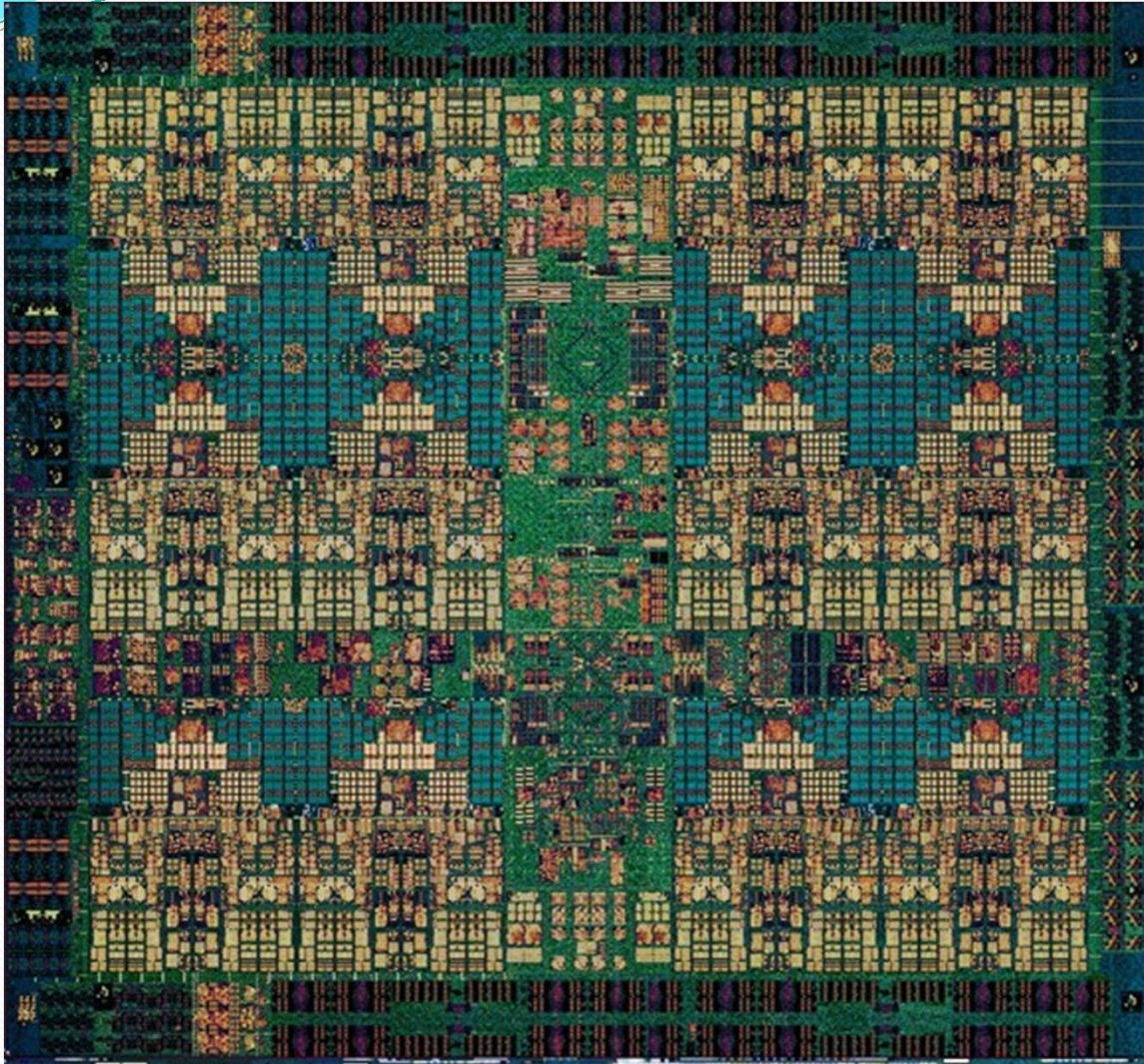


IBM Power8

- ❑ 2,75 – 4,2 ГГц
- ❑ 12 ядер x 8 нитей
Simultaneous
MultiThreading
- ❑ 64 КБ Data Cache +
32КБ instruction Cache
- ❑ L2 512 КБ
- ❑ L3 96 МБ

www.idh.ch/IBM_TU_2013/Power8.pdf

Тенденции развития современных процессоров

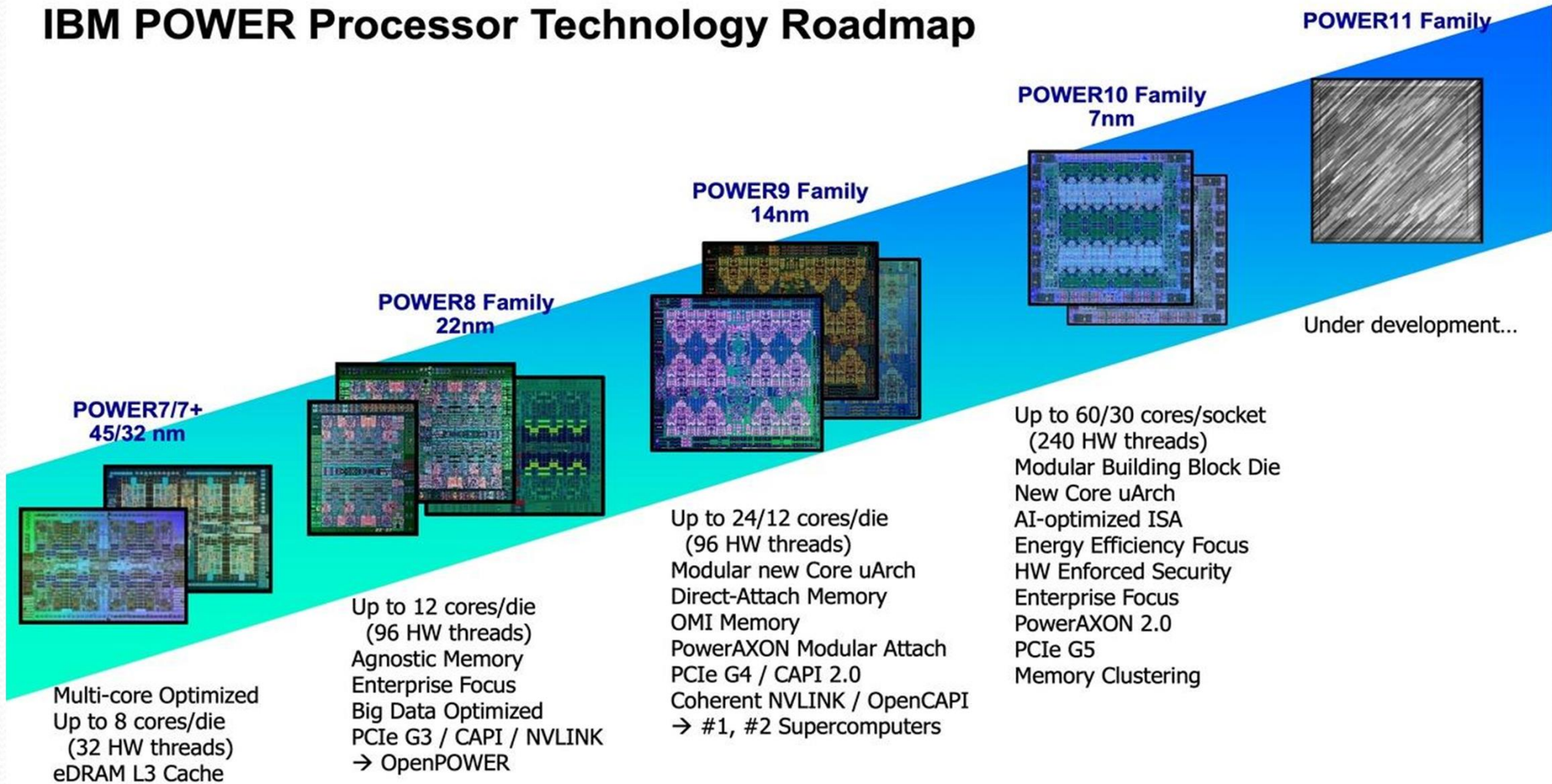


IBM Power9

- 2,75 – 4,2 ГГц
- 12 ядер x 8 нитей
24 ядра x 4 нити
- L2 512 КБ
- L3 120 МБ (10 МБ на 2 ядра)

Тенденции развития современных процессоров

IBM POWER Processor Technology Roadmap



Тенденции развития современных процессоров

POWER10 Processor Chip

Technology and Packaging:

- 602mm² 7nm Samsung (18B devices)
- 18 layer metal stack, enhanced device
- Single-chip or Dual-chip sockets

Computational Capabilities:

- Up to 15 SMT8 Cores (2 MB L2 Cache / core)
(Up to 120 simultaneous hardware threads)
- Up to 120 MB L3 cache (low latency NUCA mgmt)
- 3x energy efficiency relative to POWER9
- Enterprise thread strength optimizations
- AI and security focused ISA additions
- 2x general, 4x matrix SIMD relative to POWER9
- EA-tagged L1 cache, 4x MMU relative to POWER9

Open Memory Interface:

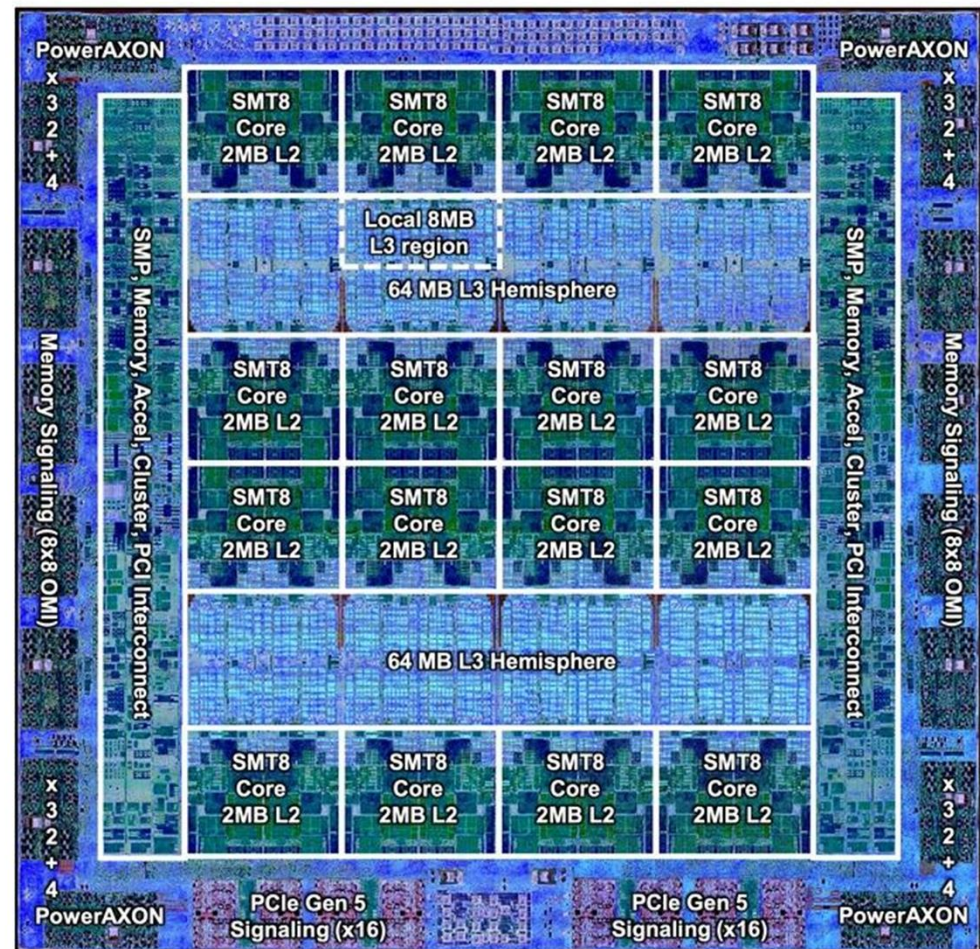
- 16 x8 at up to 32 GT/s (1 TB/s)
- Technology agnostic support: near/main/storage tiers
- Minimal (< 10ns latency) add vs DDR direct attach

PowerAXON Interface:

- 16 x8 at up to 32 GT/s (1 TB/s)
- SMP interconnect for up to 16 sockets
- OpenCAPI attach for memory, accelerators, I/O
- Integrated clustering (memory semantics)

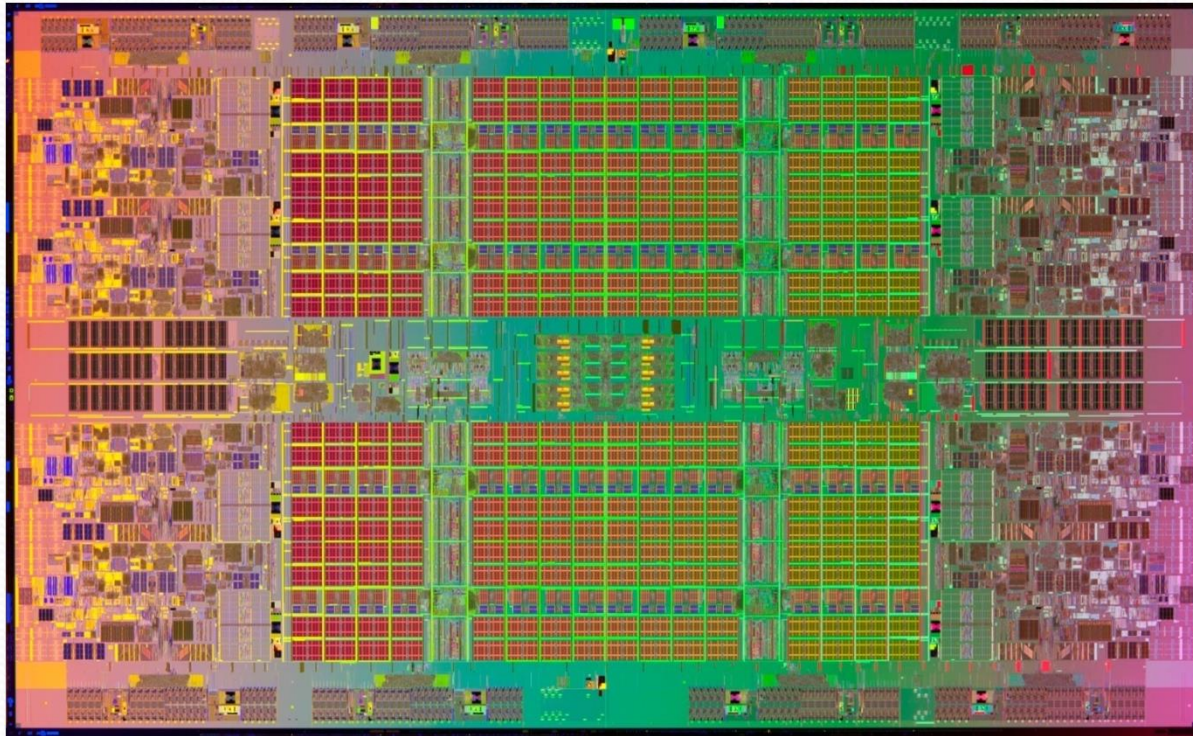
PCIe Gen 5 Interface:

- x64 / DCM at up to 32 GT/s



Die Photo courtesy of Samsung Foundry

Тенденции развития современных процессоров



Intel Itanium серии 9500

9560 8 ядер @ 2,53 ГГц, 16 нитей, 32 МБ L3 Cache

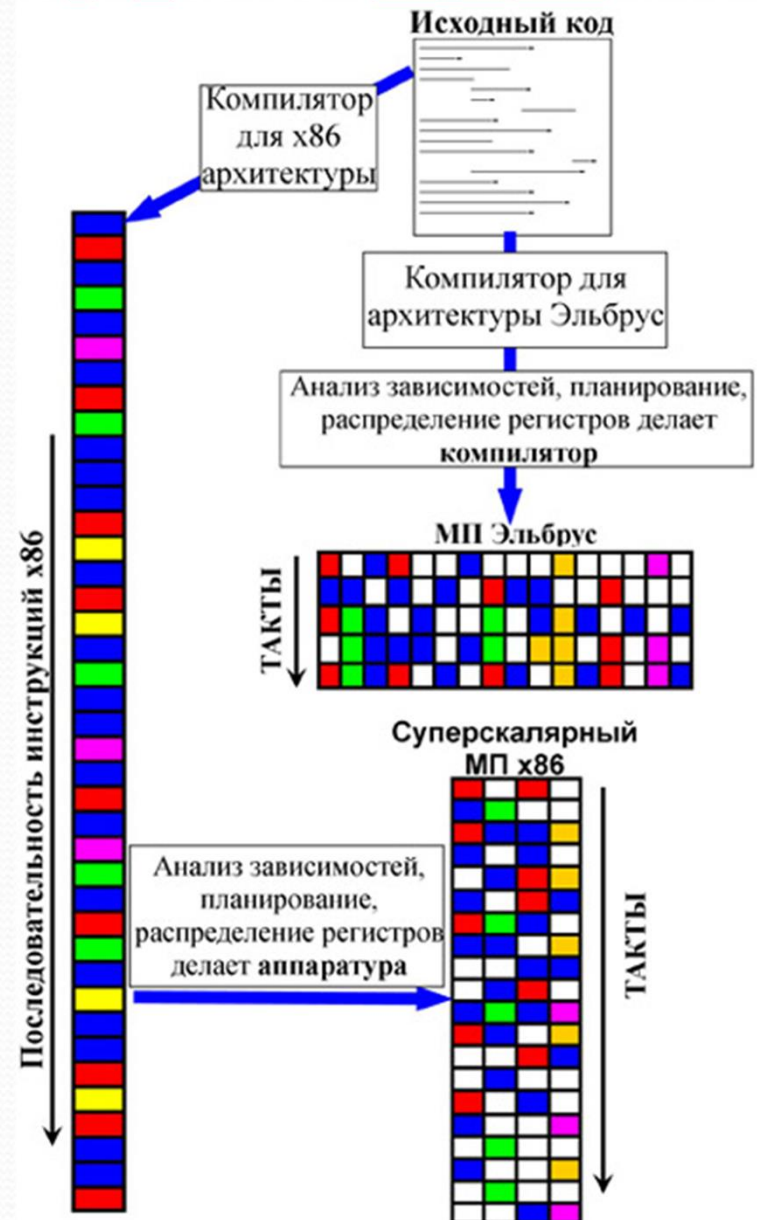
9550 4 ядра @ 2,40 ГГц, 8 нитей, 32 МБ L3 Cache

Отечественный процессор «Эльбрус-8С»



Количество ядер	8
Кэш-память 2го уровня	8 * 512 КБ
Кэш-память 3го уровня	16 МБ
Рабочая частота	1.3 ГГц
Производительность	~250 ГФлопс
Тип контроллеров памяти	DDR3-1600
Кол-во контроллеров памяти	4
Поддержка многопроцессорных систем	До 4 процессоров
Каналы межпроцессорного обмена (пропускная способность)	3 (16 ГБ/с)
Технологический процесс	28 нм
Площадь кристалла	350 кв. мм
Рассеиваемая мощность на уровне	60 – 90 Вт

Отечественный процессор «Эльбрус-8С»



Отечественный процессор «Эльбрус-16С»



Количество ядер	16
Рабочая частота	2 ГГц
Производительность	~1500 Тфлопс одинарная точность ~750 Тфлопс двойная точность
Тип контроллеров памяти	DDR4-3200
Кол-во контроллеров памяти	8
Поддержка многопроцессорных систем	До 4 процессоров
Технологический процесс	16 нм
Число транзисторов	12 млрд.



PRESENTED BY



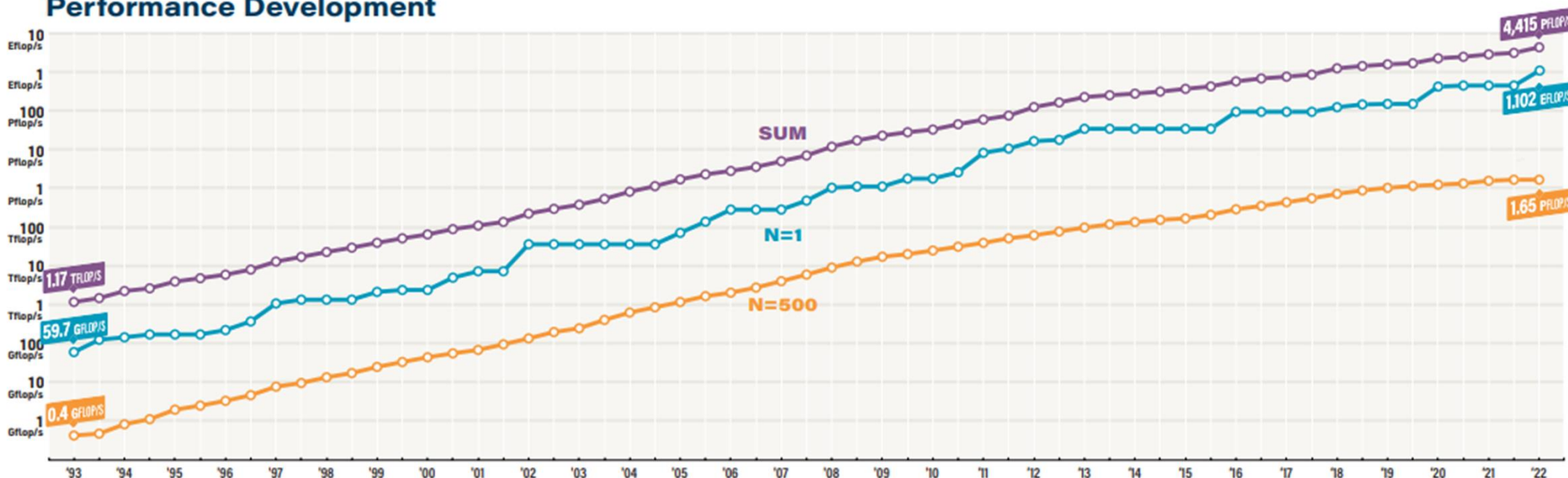
FIND OUT MORE AT

top500.org



JUNE 2022	SYSTEM	SPECS	SITE	COUNTRY	CORES	R _{MAX} PFLOP/S	POWER MW
1	Frontier	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	DOE/SC/ORNL	USA	8,730,112	1,102.0	21.3
2	Fugaku	Fujitsu A64FX (48C, 2.2GHz), Tofu Interconnect D	RIKEN R-CCS	Japan	7,630,848	442.0	29.9
3	LUMI	HPE Cray EX235a, AMD Opt 3rd Gen EPYC 64C 2GHz, AMD Instinct MI250X, Slingshot-10	EuroHPC/CSC	Finland	1,268,736	151.9	2.94
4	Summit	IBM POWER9 (22C, 3.07GHz), NVIDIA Volta GV100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/SC/ORNL	USA	2,414,592	148.6	10.1
5	Sierra	IBM POWER9 (22C, 3.1GHz), NVIDIA Tesla V100 (80C), Dual-Rail Mellanox EDR Infiniband	DOE/NNSA/LLNL	USA	1,572,480	94.6	7.44

Performance Development

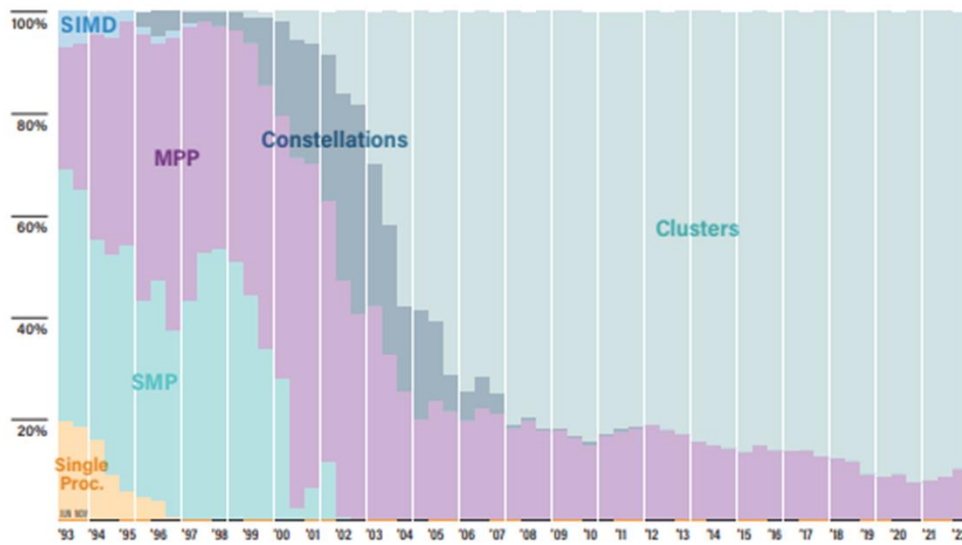


8 сентября
Москва, 2022

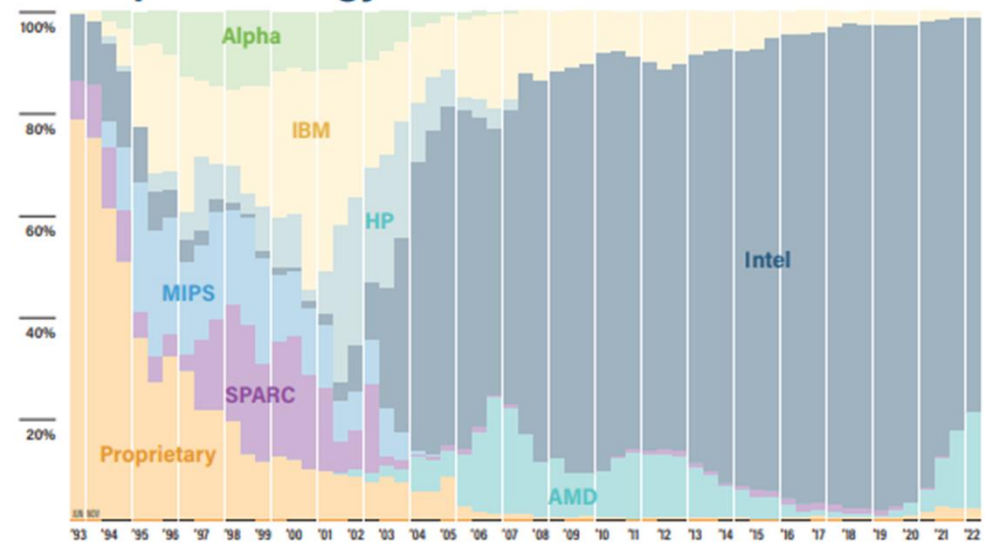
Суперкомпьютеры и параллельная обработка данных

56 из 280

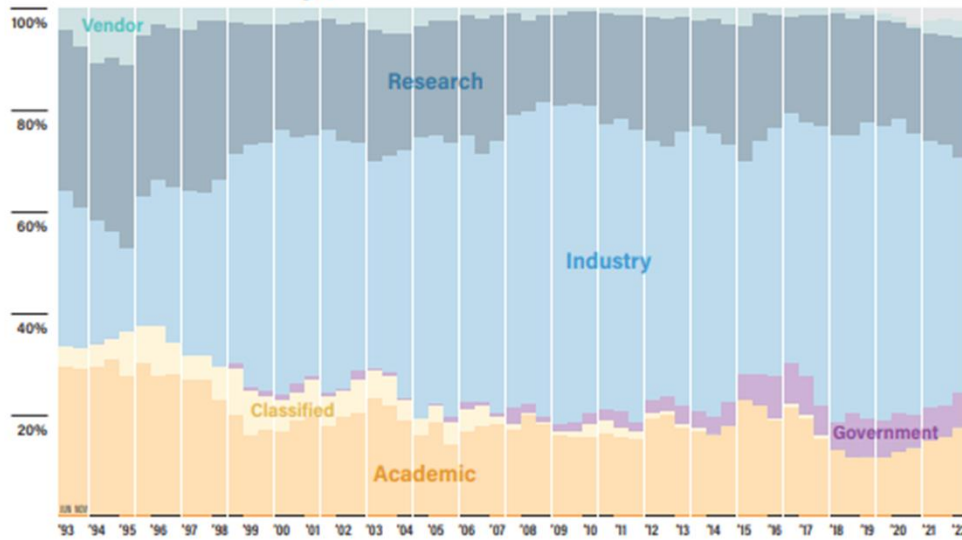
Architectures



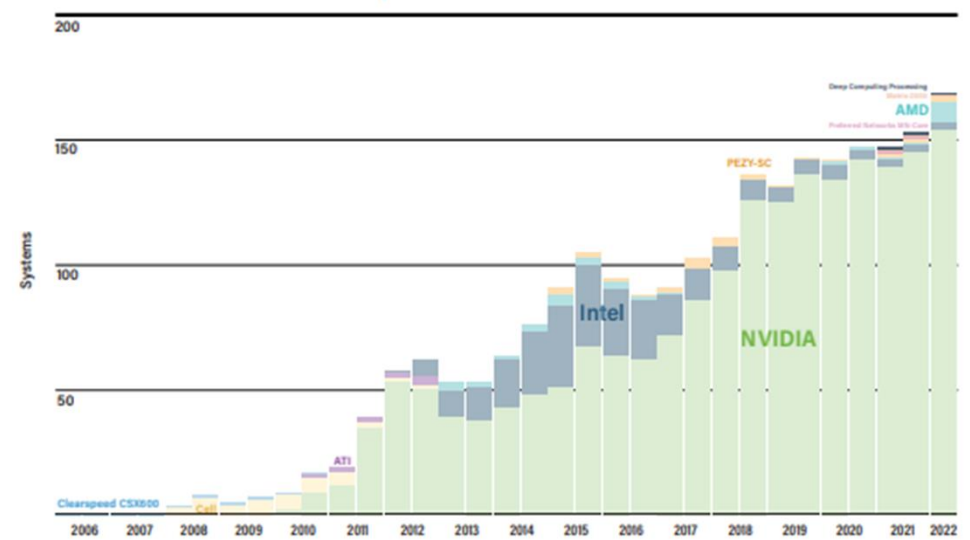
Chip Technology



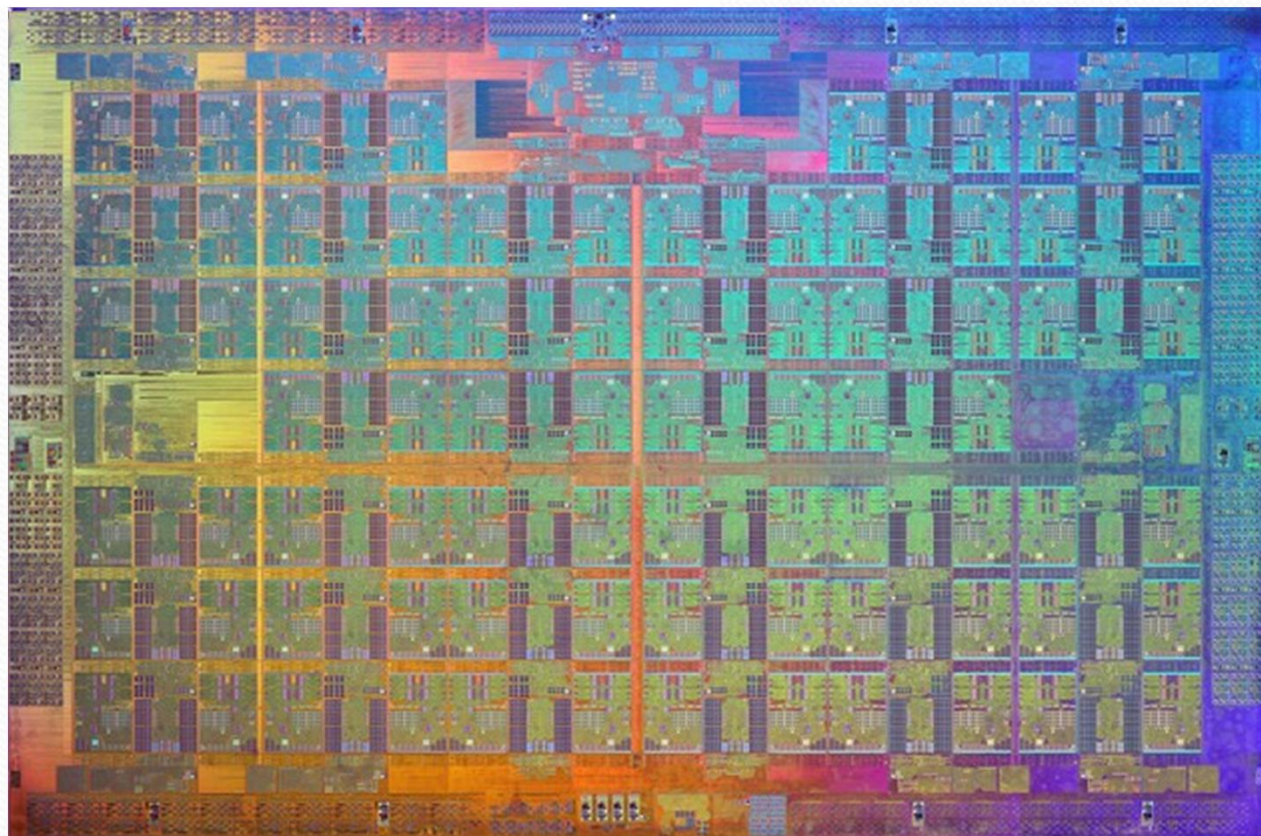
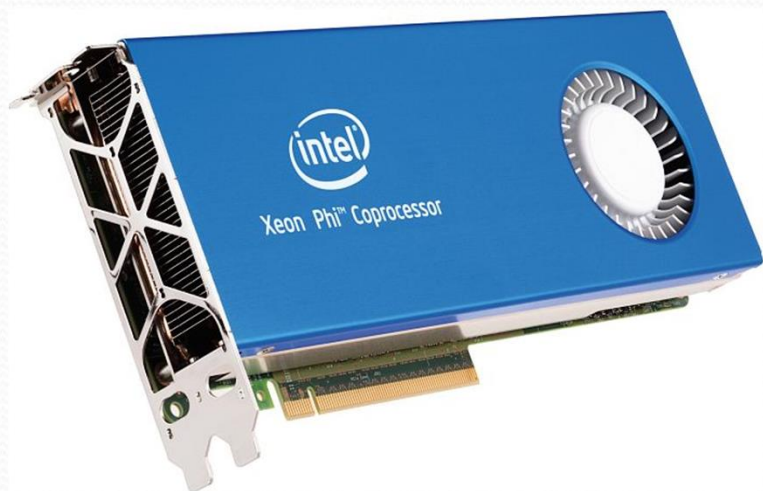
Installation Type



Accelerators/Co-processors



Intel Xeon Phi Coprocessor / Processor



8 сентября
Москва, 2022

Суперкомпьютеры и параллельная обработка данных

58 из 280

Pezy-SC Many Core Processor



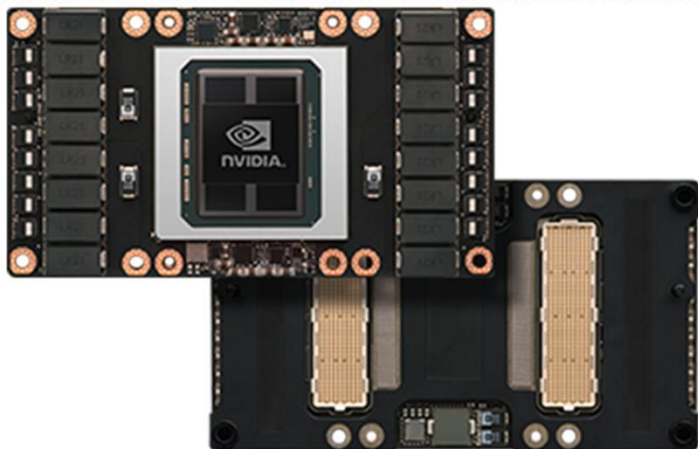
Logic Cores(PE)	1,024
Core Frequency	733MHz
Peak Performance	Floating Point Single 3.0TFlops / Double 1.5TFlops
Host Interface	PCI Express GEN3.0 x 8Lane x 4Port (x16 bifurcation available) JESD204B Protocol support
DRAM Interface	DDR4, DDR3 combo 64bit x 8Port Max B/W 1533.6GB/s +Ultra WIDE IO SDRAM (2,048bit) x 2Port Max B/W 102.4GB/s
Control CPU	ARM926 2core
Process Node	28nm
Package	FCBGA 47.5mm x 47.5mm, Ball Pitch 1mm, 2,112pin

Pezy-SC2 Many Core Processor

Logic Cores(PE)	2,048
Core Frequency	1,000MHz
Peak Performance	Half precision 16.2TFlops / Floating Point Single 8.2TFlops / Double 4.1TFlops
Host Interface	PCIe Gen3/4 x16 * 2CH (x8 * 4CH)
DRAM Interface	DDR4 64bit (ECC) * 4CH / 3,200Mbps BW=100GB/sec
CPU	MIPS64R6 (P6600) L1 I:64KB+D:64KB (each core) L2 2MB
Process Node	16 nm FinFET
Power	130 W



Графический ускоритель Nvidia P100



Конференция GTC. 19-22 сентября
<https://www.nvidia.com/gtc/>

	Tesla P100 для PCIe серверов	Tesla P100 для серверов с NVLink
Производительность операций двойной точности с плавающей точкой	4,7 Терафлопс	5,3 Терафлопс
Производительность операций одинарной точности с плавающей точкой	9,3 Терафлопс	10,6 Терафлопс
Производительность операций половинной точности с плавающей точкой	18,7 Терафлопс	21,2 Терафлопс
Пропускная способность шины NVIDIA NVLink™	-	160 ГБ/с
Пропускная способность шины PCIe x16	32 ГБ/с	32 ГБ/с
Полоса пропускания стековой памяти CoWoS с HBM2	16 ГБ или 12 ГБ	16 ГБ
Полоса пропускания стековой памяти CoWoS с HBM2	732 ГБ/с или 549 ГБ/с	732 ГБ/с
Улучшенная программируемость с технологией Page Migration Engine	✓	✓
Защита ECC для повышенной надежности	✓	✓
Оптимизация под сервер для развертывания в дата-центре	✓	✓

Графический ускоритель AMD Instinct™ MI250X



Литография	TSMC 6nm FinFET
Кол-во потоковых процессоров	14,080
Вычислительные блоки	220
Peak Engine Clock	1700 MHz
Пиковая производительность в режиме с половинной точностью (FP16)	383 TFLOPs
Peak Single Precision Matrix (FP32) Performance	95.7 TFLOPs
Peak Double Precision Matrix (FP64) Performance	95.7 TFLOPs
Пиковая производительность в режиме с одинарной точностью (FP32)	47.9 TFLOPs
Пиковая производительность в режиме с двойной точностью (FP64)	47.9 TFLOPs
Peak INT4 Performance	383 TOPs
Peak INT8 Performance	383 TOPs
Total Board Power (TBP)	500Вт 560W Peak
Dedicated Memory Size	128 ГБ
Интерфейс памяти	8192-bit
Memory Clock	1.6 GHz
Пропускная способность памяти	До 3276.8 GB/s
Память с поддержкой ECC	Да (Full-Chip)

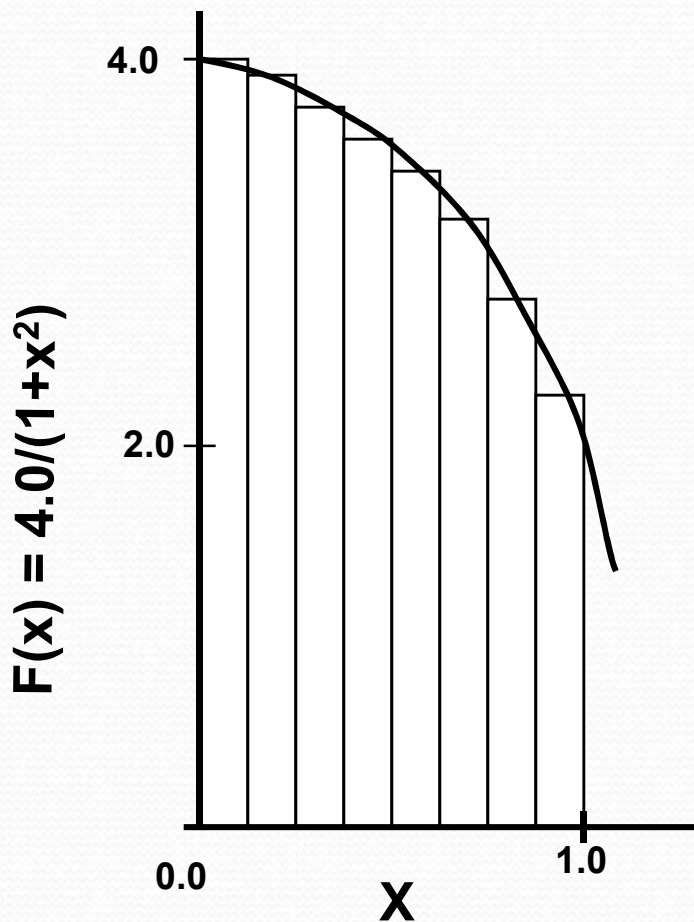
Тенденции развития современных вычислительных систем

- ❑ Темпы уменьшения латентности памяти гораздо ниже темпов ускорения процессоров + прогресс в технологии изготовления кристаллов => CMT (Chip MultiThreading)
- ❑ Опережающий рост потребления энергии при росте тактовой частоты + прогресс в технологии изготовления кристаллов => CMP (Chip MultiProcessing, многоядерность)
- ❑ И то и другое требует более глубокого распараллеливания для эффективного использования аппаратуры

Существующие подходы для создания параллельных программ для современных процессоров/систем

- ❑ Автоматическое / автоматизированное распараллеливание
- ❑ Библиотеки нитей
 - Win32 API
 - POSIX
- ❑ Библиотеки передачи сообщений
 - MPI
 - SHMEM
- ❑ OpenMP
- ❑ CUDA
- ❑ OpenACC
- ❑ DVMH

Вычисление числа π



$$\int_0^1 \frac{4.0}{(1+x^2)} dx = \pi$$

Мы можем
аппроксимировать интеграл
как сумму прямоугольников:

$$\sum_{i=0}^N F(x_i) \Delta x \approx \pi$$

Где каждый прямоугольник
имеет ширину Δx и высоту
 $F(x_i)$ в середине интервала

Вычисление числа π . Последовательная программа

```
#include <stdio.h>
int main ()
{
    int n =100000, i;
    double pi, h, sum, x;
    h = 1.0 / (double) n;
    sum = 0.0;
    for (i = 1; i <= n; i ++)
    {
        x = h * ((double)i - 0.5);
        sum += (4.0 / (1.0 + x*x));
    }
    pi = h * sum;
    printf("pi is approximately %.16f", pi);
    return 0;
}
```

Автоматическое распараллеливание

Polaris, CAPO, WPP, SUIF, VAST/Parallel, OSCAR, Intel/OpenMP, ParaWise

```
icc -parallel pi.c
```

```
pi.c(8): (col. 5) remark: LOOP WAS AUTO-PARALLELIZED.
```

```
pi.c(8): (col. 5) remark: LOOP WAS VECTORIZED.
```

```
pi.c(8): (col. 5) remark: LOOP WAS VECTORIZED.
```

В общем случае, автоматическое распараллеливание затруднено:

- ❑ косвенная индексация ($A[B[i]]$);
- ❑ указатели (ассоциация по памяти);
- ❑ сложный межпроцедурный анализ.

Автоматизированное распараллеливание

Intel/GAP (Guided Auto-Parallel), CAPTools/ParaWise, BERT77, FORGE Magic/DM, ДВОР (Диалоговый Высокоуровневый Оптимизирующий Распараллеливатель), САПФОР (Система Автоматизации Параллелизации ФОРтран программ)

```
for (i=0; i<n; i++) {  
    if (A[i] > 0) {b=A[i]; A[i] = 1 / A[i]; }  
    if (A[i] > 1) {A[i] += b;}  
}
```

```
icc -guide -parallel test.cpp
```

Автоматизированное распараллеливание

test.cpp(49): remark #30521: (PAR) Loop at line 49 cannot be parallelized due to conditional assignment(s) into the following variable(s): b. This loop will be parallelized if the variable(s) become unconditionally initialized at the top of every iteration. [VERIFY] Make sure that the value(s) of the variable(s) read in any iteration of the loop must have been written earlier in the same iteration.

test.cpp(49): remark #30525: (PAR) If the trip count of the loop at line 49 is greater than 188, then use "#pragma loop count min(188)" to parallelize this loop. [VERIFY] Make sure that the loop has a minimum of 188 iterations.

```
#pragma loop count min (188)
for (i=0; i<n; i++) {
    b = A[i];
    if (A[i] > 0) {A[i] = 1 / A[i];}
    if (A[i] > 1) {A[i] += b;}
}
```

Вычисление числа π с использованием Win32 API

```
#include <stdio.h>
#include <windows.h>
#define NUM_THREADS 2
CRITICAL_SECTION hCriticalSection;
double pi = 0.0;
int n = 100000;
void main ()
{
    int i, threadArg[NUM_THREADS];
    DWORD threadID;
    HANDLE threadHandles[NUM_THREADS];
    for(i=0; i<NUM_THREADS; i++) threadArg[i] = i+1;
    InitializeCriticalSection(&hCriticalSection);
    for (i=0; i<NUM_THREADS; i++) threadHandles[i] =
        CreateThread(0,0,(LPTHREAD_START_ROUTINE) Pi,&threadArg[i], 0, &threadID);
    WaitForMultipleObjects(NUM_THREADS, threadHandles, TRUE,INFINITE);
    printf("pi is approximately %.16f", pi);
}
```

Вычисление числа π с использованием Win32 API

```
void Pi (void *arg)
{
    int i, start;
    double h, sum, x;
    h = 1.0 / (double) n;
    sum = 0.0;
    start = *(int *) arg;
    for (i=start; i<= n; i=i+NUM_THREADS)
    {
        x = h * ((double)i - 0.5);
        sum += (4.0 / (1.0 + x*x));
    }
    EnterCriticalSection(&hCriticalSection);
    pi += h * sum;
    LeaveCriticalSection(&hCriticalSection);
}
```

Взаимное исключение критических интервалов

При взаимодействии через общую память нити должны синхронизовать свое выполнение.

Thread0: $pi = pi + val$; && Thread1: $pi = pi + val$;

Время	Thread 0	Thread 1
1	LOAD R1,pi	
2	LOAD R2,val	
3	ADD R1,R2	LOAD R3,pi
4	STORE R1,pi	LOAD R4,val
5		ADD R3,R4
6		STORE R3,pi

Результат зависит от порядка выполнения команд. Требуется взаимное исключение критических интервалов.

Вычисление числа π с использованием OpenMP

```
#include <stdio.h>
int main ()
{
    int n =100000, i;
    double pi, h, sum, x;
    h = 1.0 / (double) n;
    sum = 0.0;
    #pragma omp parallel for reduction(+:sum) private(x)
    for (i = 1; i <= n; i ++)
    {
        x = h * ((double)i - 0.5);
        sum += (4.0 / (1.0 + x*x));
    }
    pi = h * sum;
    printf("pi is approximately %.16f", pi);
    return 0;
```

Вычисление числа π с использованием MPI

```
#include "mpi.h"
#include <stdio.h>
int main (int argc, char *argv[])
{
    int n =100000, myid, numprocs, i;
    double mypi, pi, h, sum, x;
    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    h = 1.0 / (double) n;
    sum = 0.0;
```

Вычисление числа π с использованием MPI

```
for (i = myid + 1; i <= n; i += numprocs)
{
    x = h * ((double)i - 0.5);
    sum += (4.0 / (1.0 + x*x));
}
mypi = h * sum;
MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);
if (myid == 0) printf("pi is approximately %.16f", pi);
MPI_Finalize();
return 0;
}
```

Вычисление числа π с использованием SHMEM

```
#include <shmem.h>
#include <stdio.h>
long sync[SHMEM_REDUCE_SYNC_SIZE] = {SHMEM_SYNC_VALUE};
double work[SHMEM_REDUCE_MIN_WRKDATA_SIZE];
double pi;
int main (int argc, char *argv[])
{
    int n =100000, myid, numprocs, i;
    double h, sum, x;
    shmem_init();
    numprocs = shmem_n_pes();
    myid = shmem_my_pe();
    h = 1.0 / (double) n;
    sum = 0.0;
```

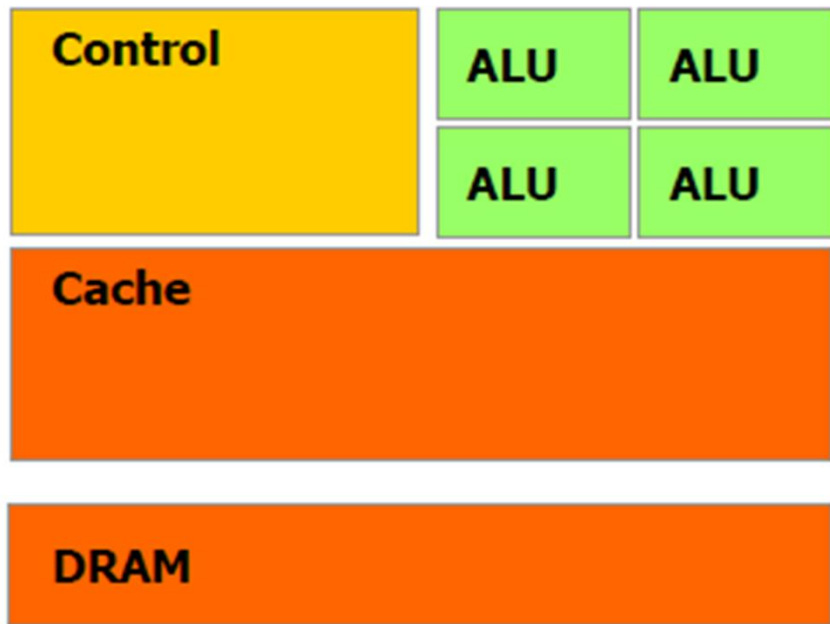
Вычисление числа π с использованием SHMEM

```
for (i = myid + 1; i <= n; i += numprocs)
{
    x = h * ((double)i - 0.5);
    sum += (4.0 / (1.0 + x*x));
}
pi = h * sum;
shmem_double_sum_to_all(&pi, &pi, 1, 0, 0, numprocs, work, sync);
if (myid == 0) printf("pi is approximately %.16f", pi);
shmem_finalize();
return 0;
}
```

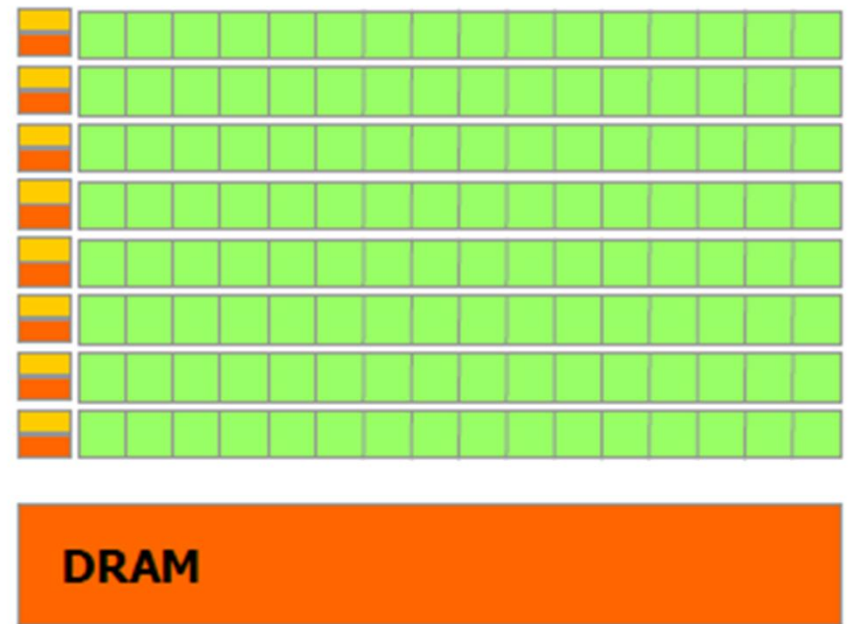
Вычисление числа π с использованием UPC

```
#include <stdio.h>
#include <upc.h>
#include <upc_collective.h>
shared double pi;
shared double sum[THREADS];
int main ()
{
    int n =100000, i;
    double h, x;
    h = 1.0 / (double) n;
    sum[MYTHREAD] = 0.0;
    for (i=1+MYTHREAD; i<=n;i+= THREADS)
    {
        x = h * ((double)i - 0.5);
        sum[MYTHREAD] += (4.0 / (1.0 + x*x));
    }
    upc_all_reducel (&pi, sum, UPC_ADD,
        THREADS, 1,NULL,
        UPC_IN_ALLSYNC | UPC_OUT_ALLSYNC);
    if (MYTHREAD == 0)
    {
        pi *= h;
        printf("pi is approximately %.16f", pi);
    }
    return 0;
}
```

CPU или GPU



CPU



GPU

Вычисление числа π с использованием CUDA

```
#include <stdio.h>
#include <cuda.h>
#define N 1000000
#define NUM_BLOCK 32 // Number of thread blocks
#define NUM_THREAD 32 // Number of threads per block
int tid;
float pi = 0;
// Kernel that executes on the CUDA device
__global__ void cal_pi(float *sum) {
    int i;
    float x, step=1.0/N; // Step size
    // Sequential thread index across the blocks
    int idx = blockIdx.x*blockDim.x+threadIdx.x;
    for (i=idx; i< N; i+=NUM_BLOCK*NUM_THREAD) {
        x = (i+0.5)*step;
        sum[idx] += 4.0/(1.0+x*x);
    }
}
```

<http://cacs.usc.edu/education/cs596/src/cuda/pi.cu>

Вычисление числа π с использованием CUDA

```
int main(void) { // Main routine that executes on the host
    dim3 dimGrid(NUM_BLOCK,1,1); // Grid dimensions
    dim3 dimBlock(NUM_THREAD,1,1); // Block dimensions
    float *sumHost, *sumDev; // Pointer to host & device arrays
    size_t size = NUM_BLOCK*NUM_THREAD*sizeof(float); //Array size
    sumHost = (float *)malloc(size); // Allocate array on host
    cudaMalloc((void **) &sumDev, size); // Allocate array on device
    cudaMemset(sumDev, 0, size); // Initialize array in device to 0
    // Do calculation on device
    cal_pi <<<dimGrid, dimBlock>>> (sumDev); // call CUDA kernel
    // Retrieve result from device and store it in host array
    cudaMemcpy(sumHost, sumDev, size, cudaMemcpyDeviceToHost);
    for(tid=0; tid<NUM_THREAD*NUM_BLOCK; tid++) pi += sumHost[tid];
    pi *= step;
    printf("PI = %f\n",pi); // Print results
    free(sumHost); // Cleanup
    cudaFree(sumDev);
    return 0;
```

<http://cacs.usc.edu/education/cs596/src/cuda/pi.cu>

Вычисление числа π с использованием OpenACC

```
#include <stdio.h>
int main ()
{
    int n =100000, i;
    double pi, h, sum, x;
    h = 1.0 / (double) n;
    sum = 0.0;
    #pragma acc parallel loop
    for (i = 1; i <= n; i ++)
    {
        x = h * ((double)i - 0.5);
        sum += (4.0 / (1.0 + x*x));
    }
    pi = h * sum;
    printf("pi is approximately %.16f", pi);
    return 0;
}
```

```
pgcc -acc test.c -Minfo=all
main:
    8, Accelerator kernel generated
    Generating Tesla code
    9, #pragma acc loop gang, vector(128) /* blockIdx.x
    threadIdx.x */
    12, Sum reduction generated for sum
```

Вычисление числа π с использованием DVM

```
#include <stdio.h>
int main ()
{
    int n =100000, i;
    double pi, h, s, x;
    #pragma dvm template[n] distribute[block]
    void *tmp;
    h = 1.0 / (double) n;
    s = 0.0;
    #pragma dvm parallel (i on tmp[i]) reduction(sum(s)) private(x)
    for (i = 1; i <= n; i ++)
    {
        x = h * ((double)i - 0.5);
        s += (4.0 / (1.0 + x*x));
    }
    pi = h * s;
    printf("pi is approximately %.16f", pi);
    return 0;
}
```

Вычисление числа π с использованием OpenMP

```
#include <stdio.h>
int main ()
{
    int n =100000, i;
    double pi, h, sum, x;
    h = 1.0 / (double) n;
    sum = 0.0;
    #pragma omp parallel for reduction(+:sum) private(x)
    for (i = 1; i <= n; i ++)
    {
        x = h * ((double)i - 0.5);
        sum += (4.0 / (1.0 + x*x));
    }
    pi = h * sum;
    printf("pi is approximately %.16f", pi);
    return 0;
```

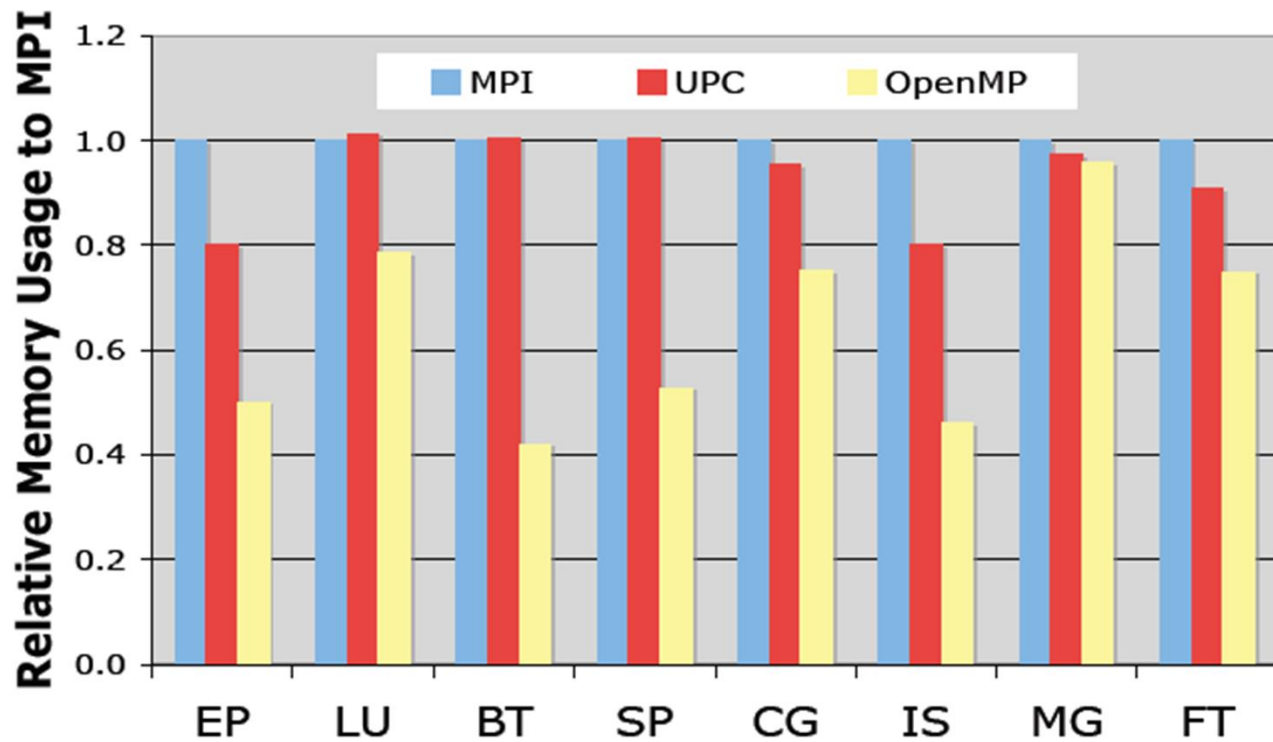
Достоинства использования OpenMP вместо MPI для многоядерных процессоров

- ❑ Возможность инкрементального распараллеливания
- ❑ Упрощение программирования и эффективность на нерегулярных вычислениях, проводимых над общими данными
- ❑ Ликвидация дублирования данных в памяти, свойственного MPI-программам
- ❑ Объем памяти пропорционален быстродействию процессора. В последние годы увеличение производительности процессора достигается удвоением числа ядер, при этом частота каждого ядра снижается. Наблюдается тенденция к сокращению объема оперативной памяти, приходящейся на одно ядро. Присущая OpenMP экономия памяти становится очень важна.
- ❑ Наличие локальных и/или разделяемых ядрами КЭШей будут учитываться при оптимизации OpenMP-программ компиляторами, что не могут делать компиляторы с последовательных языков для MPI-процессов.

Тесты NAS

BT	3D Навье-Стокс, метод переменных направлений
CG	Оценка наибольшего собственного значения симметричной разреженной матрицы
EP	Генерация пар случайных чисел Гаусса
FT	Быстрое преобразование Фурье, 3D спектральный метод
IS	Параллельная сортировка
LU	3D Навье-Стокс, метод верхней релаксации
MG	3D уравнение Пуассона, метод Multigrid
SP	3D Навье-Стокс, Beam-Warning approximate factorization

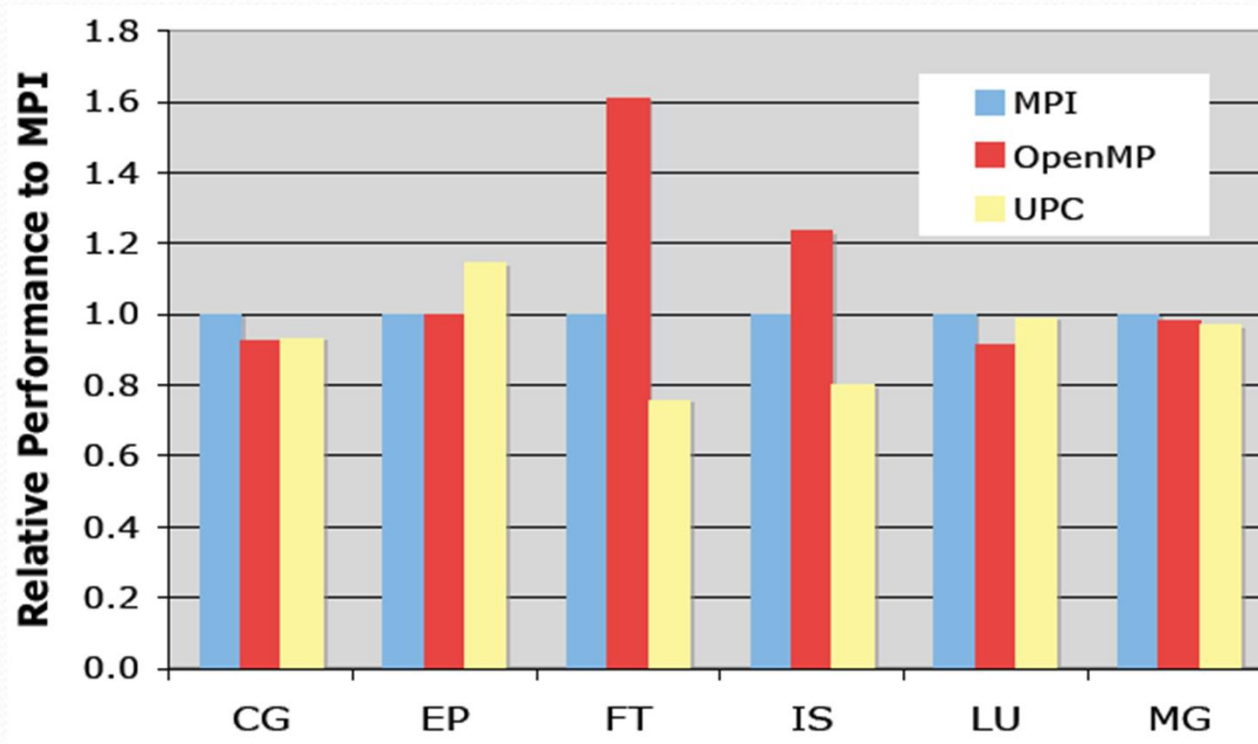
Тесты NAS



Analyzing the Effect of Different Programming Models Upon Performance and Memory Usage on Cray XT5 Platforms

<https://www.nersc.gov/assets/NERSC-Staff-Publications/2010/Cug2010Shan.pdf>

Тесты NAS



Analyzing the Effect of Different Programming Models Upon Performance and Memory Usage on Cray XT5 Platforms

<https://www.nersc.gov/assets/NERSC-Staff-Publications/2010/Cug2010Shan.pdf>

Содержание

- ❑ Тенденции развития современных вычислительных систем
- ❑ OpenMP – модель параллелизма по управлению
- ❑ Конструкции распределения работы
- ❑ Конструкции для синхронизации нитей
- ❑ Система поддержки выполнения OpenMP-программ
- ❑ Новые возможности OpenMP