Введение в параллельные вычисления

КС-40, КС-44
РХТУ

Преподаватель
Митричев Иван Игоревич, к.т.н.

2017

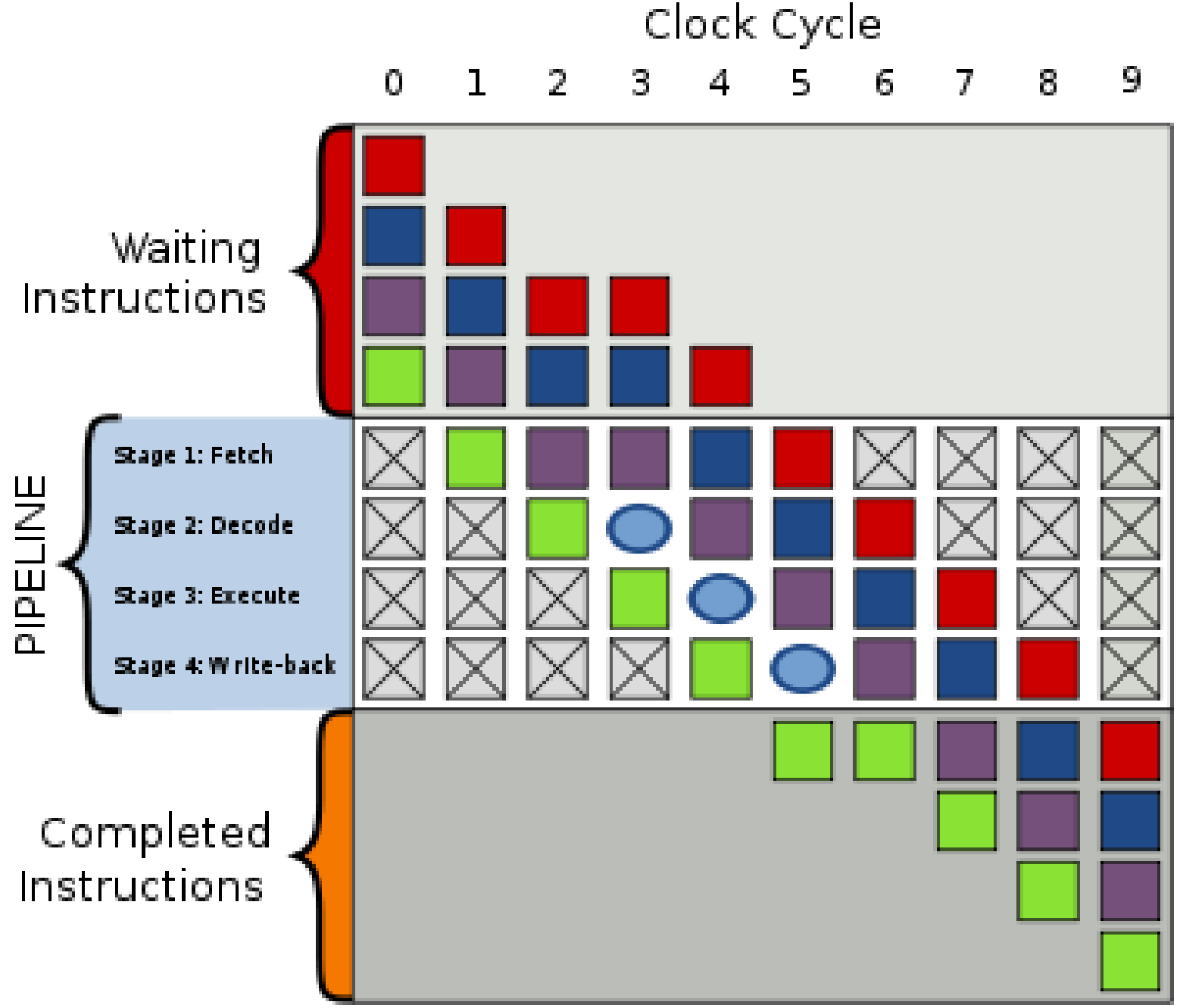
Параллелизация

Параллелизация на уровне программного обеспечения

Параллелизация на уровне оборудования

- конвейерная обработка
- увеличение количества функциональных узлов процессора (суперскалярность)
- увеличение количества ядер (многоядерность)
- увеличение количества процессоров (многопроцессорность)

Конвейерная обработка



Из-за задержки исполнения фиолетовой инструкции - простой конвейера

Классификация архитектуры процессоров

RISC - быстродействие повышается за счёт упрощенного набора инструкций.

+ Повышение частоты

CISC – расширенный набор команд

Современные процессоры Intel основаны на архитектуре CISC с RISC-ядром.

VLIW – распределение компилятором команд между вычислительными устройствами.

+ Упрощает процессор, снижает энергопотребление (Qualcomm Snapdragon)

- Пустые инструкции для простаивающих устройств повышают размер программ

Классификация Флинна

60-70е годы
SISD (single instruction stream, single data stream — один поток инструкций, один поток данных) - скалярные процессоры
SIMD (single instruction stream, multiple data streams — один поток инструкций, несколько потоков данных) - процессоры с векторизацией
MIMD (multiple instruction streams, multiple data streams — несколько потоков инструкций, несколько потоков данных) - многоядерные процессоры
MISD - не создавались

В настоящее время устарела (почти все - MIMD)

Общая и распределенная память

Современные многопроцессорные системы:

- системы с общей памятью (англ. shared memory, SM)
- системы с распределенной памятью (англ. distributed memory, DM)
- системы с распределенной общей памятью (англ. distributed shared memory, DSM)

Архитектура современных многопроцессорных систем

SMP - симметричная мультипроцессорность (общая память). *Технологии программирования OpenMP, C++ Threads*

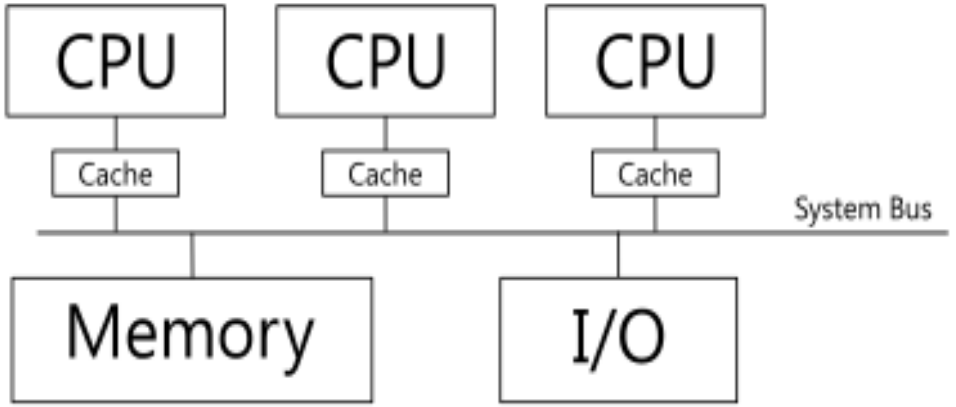
NUMA – неоднородный доступ к памяти (Non-Uniform Memory Access)

MPP – массивно-параллельная обработка

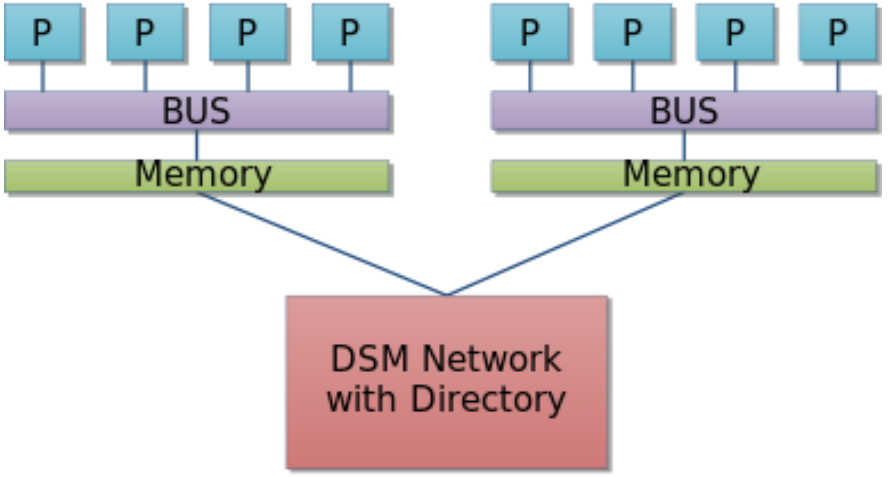
- ✓ Система строится из отдельных узлов (node). Обмен данными за счет передачи сообщений
  - снижается скорость доступа к данным
  + повышается масштабируемость
- ✓ *Технология программирования MPI*

Системы с общей памятью

SMP



NUMA

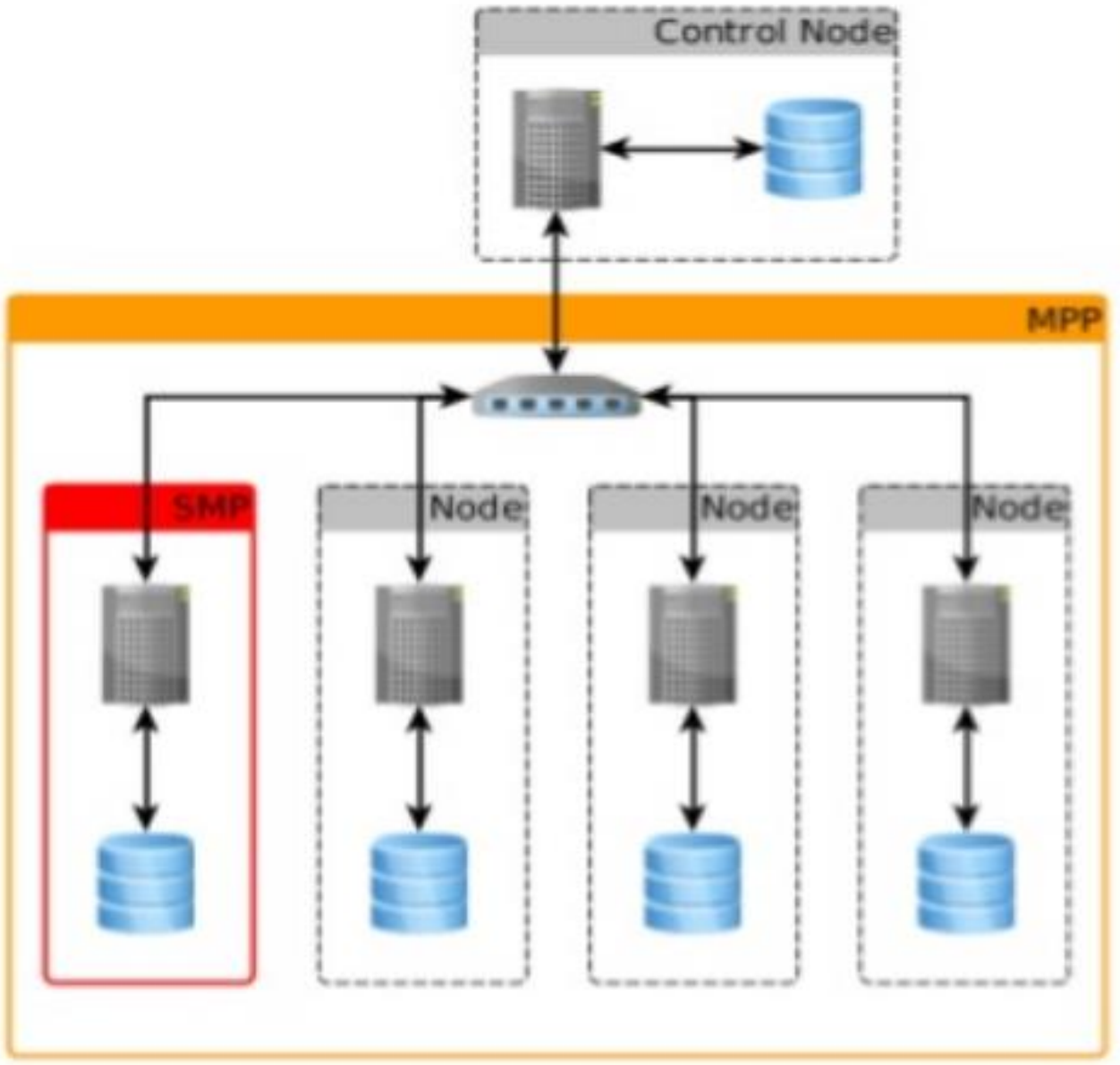


Проблема когерентности кэша - cc-NUMA
(Cache Coherent Non-Uniform Memory Access)

системы с общей памятью плохо
масштабируются

Системы с распределенной памятью

MPP



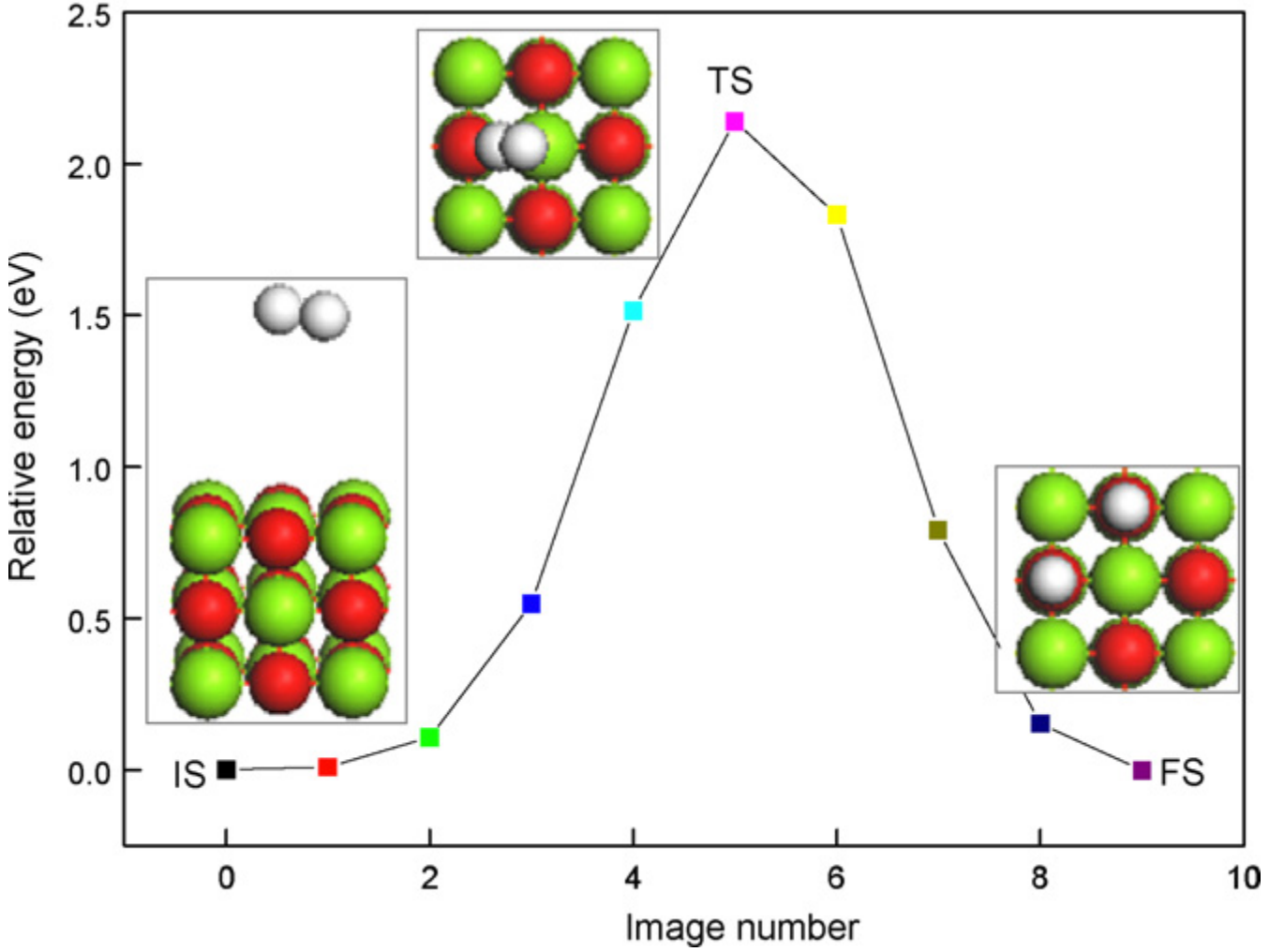
Параллельные вычисления в современном мире

Пример: Квантовая химия (Quantum Espresso).

Расчет переходного состояния реакции, метод Nudged Elastic Band

...уровни параллелизации...
Изображения (images) - снимки пространственной конфигурации реагирующей системы в различные моменты времени
Пулы (pools) - группа процессов, рассчитывающая заданные к-точки
PW - группа процессов, рассчитывающая заданный набор орбиталей
Задания (tasks) - распараллеливание 3D-преобразования Фурье
Группа линейной алгебры (linear-algebra group) - независимый уровень распараллеливания для операций линейной алгебры (работа с матрицами)

Диссоциация молекулы водорода на поверхности MgO, NEB



Wu G. et al. Adsorption and dissociation of hydrogen on MgO surface: A first-principles study //Journal of Alloys and Compounds. – 2009. – Т. 480. – №. 2. – С. 788-793.
Available at: https://www.researchgate.net/publication/222953851_Adsorption_and_dissociation_of_hydrogen_on_MgO_surface_A_first-principles_study

Как измеряют производительность компьютеров?

• Единица измерений Flops (флопс, – операций с плавающей точкой в
секунду).
• Пиковая производительность
• Реальная производительность
• Бенчмарк – эталонная тестовая программа для оценки производительности.
• High Performance Linpack – тест, основанный на задаче решения системы линейных алгебраических уравнений.
• В настоящее время производительность наиболее мощных компьютеров превысила десятки петафлопсов (10^{15} Флопс)

Пиковая производительность

Процессор Intel Core 2 является суперскалярным и содержит 2 устройства вычислений, выполняющих по 2 операции за такт.

Для процессора, имеющего в своём составе 4 ядра (Core 2 Quad) и работающего на частоте 3,5 ГГц, теоретический предел производительности составляет 4 x 4 x 3,5 = 56 гигафлопсов

идеальные условия. Нужна реальная метрика

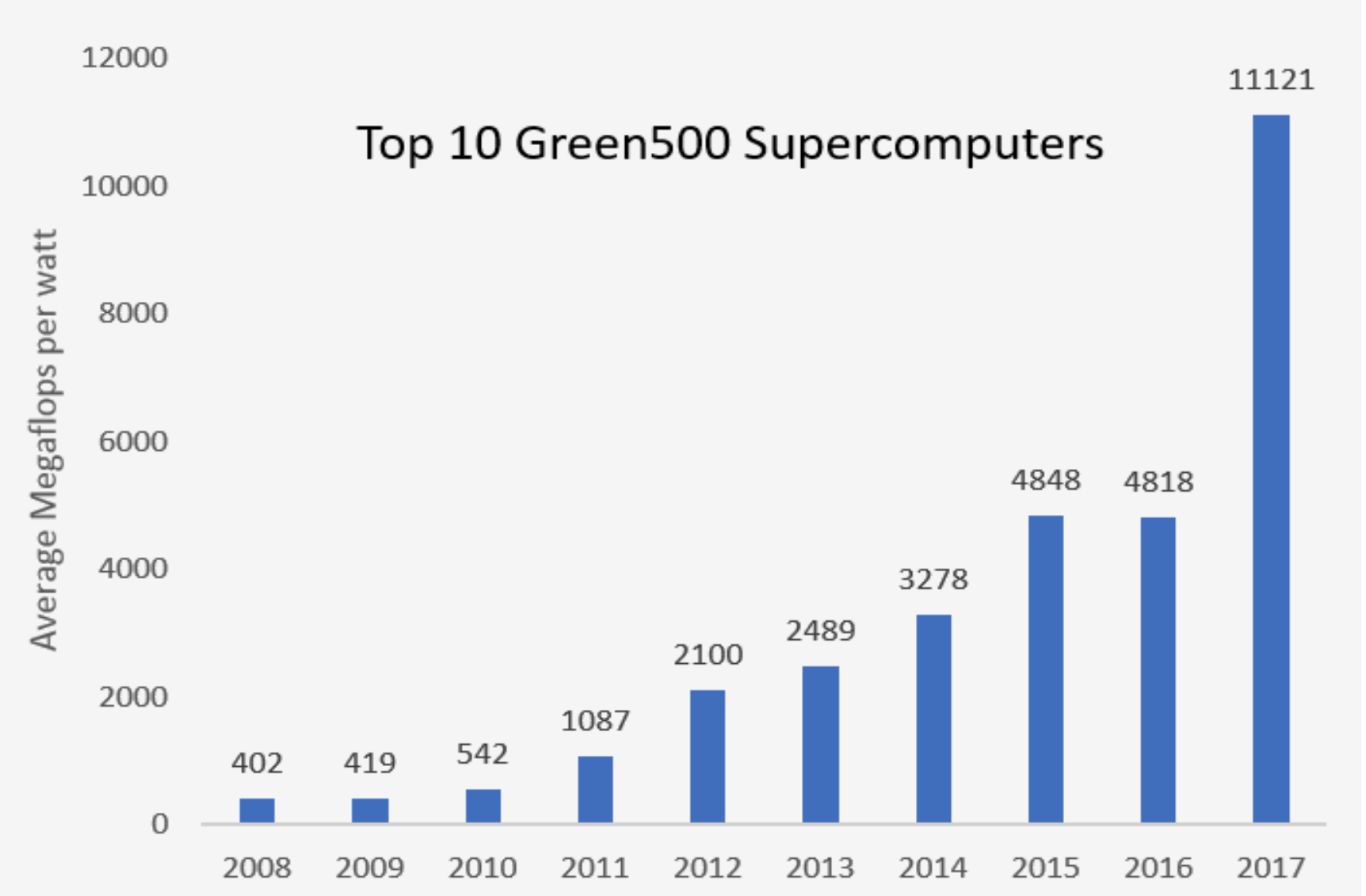
Реальная производительность на LINPACK

Рейтинг - ТОП-500

https://www.top500.org/lists/2017/06/

Наиболее мощный в мире компьютер
Sunway TaihuLight
установлен в Государственном вычислительном центре, Wuxi, Китай
Ядер: 10649600, Память: 1310720 GB,
Интерконнект: Sunway, Процессоры: Sunway SW26010 260C 1.45GHz
Макс. теор. произв., Пфлопс: 125.435
Производительность на тестах Linpack, Пфлопс: 93.0146
Потребляемая мощность 15371 КВт!!!

Энергосбережение: "Green-500"



А что в России?

суперкомпьютер Ломоносов (НИВЦ МГУ)

Ломоносов-2 (НИВЦ МГУ)

Гибридная архитектура

Закон Мура

1965, 1975 г., Гордон Мур:
«количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 2 года»

2007 г. – предсказано отклонение от закона в будущем - ясен предел (атомный уровень)

Рост производительности за счет параллелизма

Ограничения параллельных вычислений: солдаты и поле

На одном поле заданного размера оптимальное число работающих солдат является конечным

Кластерные системы

Кластер – группа компьютеров, объединенных высокоскоростным сетевым соединением, которую можно использовать как единый вычислительный ресурс.

В отличие от суперкомпьютеров собираются из простых деталей.

- Легко масштабируется.
- Управляется обычной операционной системой Linux.
- Имеет простую коммуникационную среду.
- Узлы кластера могут быть однопроцессорными и многопроцессорными.
- Кластер может быть создан на основе компьютерного класса.
- Кластер может быть собран в стойку из серверов - лезвий.

Закон Амдала

Позволяет оценить ускорение S, которое может быть получено на компьютере из p процессоров при данном значении α – доли (по времени) операций, которые нужно выполнять последовательно, 0 <= α <=1

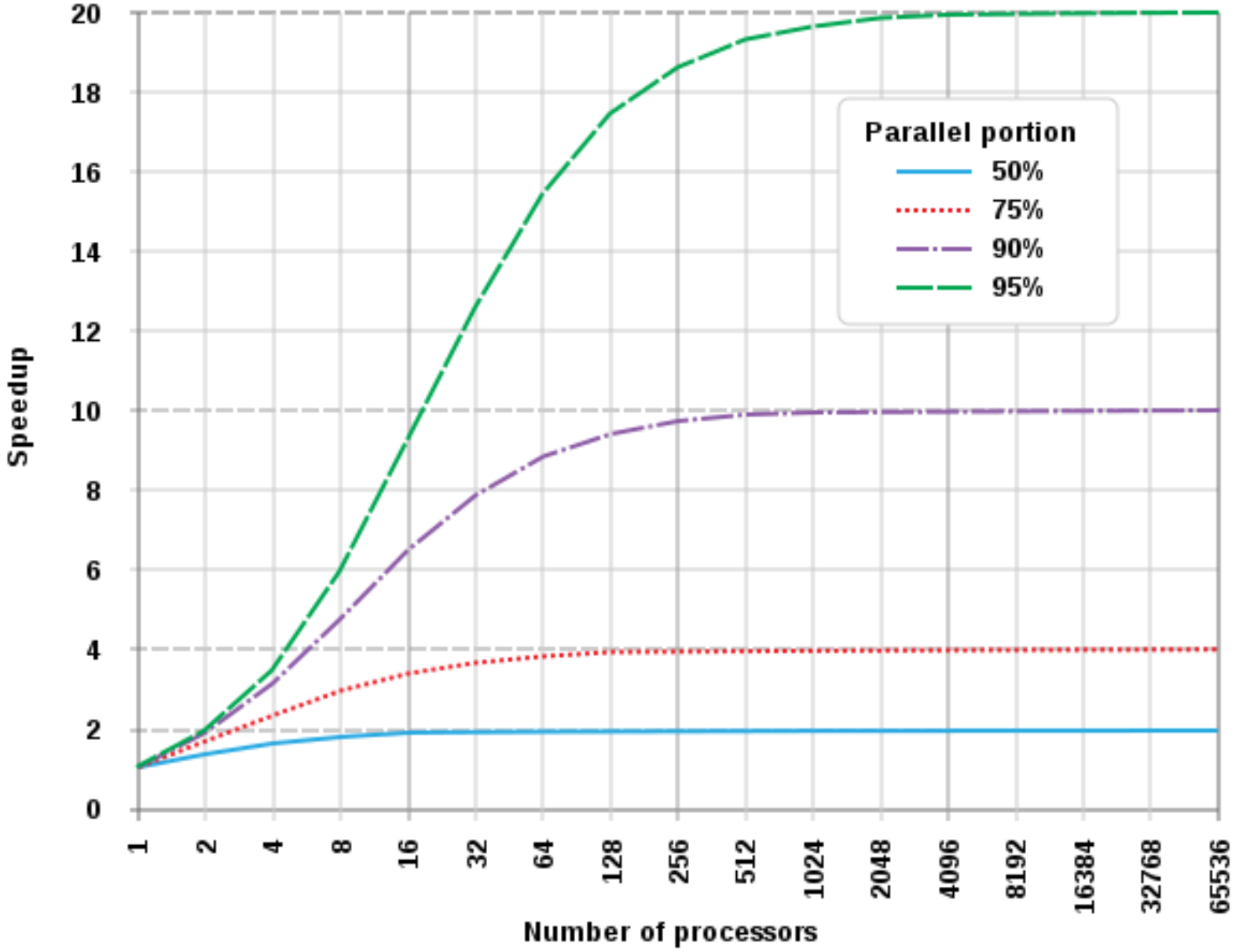
$$S_p = \frac{1}{\alpha + \frac{1-\alpha}{p}}$$

Закон Амдала: пример

95% времени работы программы – инструкции, которые можно выполнять параллельно.
Какое максимальное ускорение можно получить при параллельном выполнении
по сравнению с последовательным выполнением программы?

$$S \leq \frac{1}{0.05 + (1 - 0.05)/8} \cong 5.9$$

Закон Амдала



Linux и потоки

Linux поддерживает многозадачность и многопоточность, т.е., в системе одновременно может работать несколько задач (процессов), и каждая из задач может выполнятся в несколько потоков.

Поток (thread) – минимальная часть процесса, исполнение которой назначается ОС.

Могут выполняться на различных ядрах.
Используется переключение контекста для выбора потоков, исполняемых на одном ядре (как и в случае процессов).

Потоки

Процесс	Поток
Независим	Составной элемент процесса
Имеет собственное адресное пространство (память)	Разделяет адресное пространство с другими потоками
Взаимодействуют через специальные механизмы ОС (file, signal, socket, pipe, semaphore, shared memory)	Данные являются общими, могут изменяться напрямую
Переключение контекста и порождение более времязатратны	Выбор (переключение контекста) и порождение происходят быстрее

Сколько можно создать потоков?

- cat /proc/sys/kernel/threads-max
  - обычно, 30-150 тысяч

Сколько можно создать процессов?

- cat /proc/sys/kernel/pid_max
  - обычно, 32767

Зачем нужны многопоточные программы

1) Независимое использование интерфейса и вычислительной части программы
2) Улучшение времени реакции интерактивных программ (фоновое скачивание, проверка орфографии)
3) Улучшение времени реакции серверных приложений
4) Параллелизация кода для уменьшения времени работы программы (на SMP-системах)

Работа с потоками на Linux

#include <thread>

Eclipse:
Установить опции --std=c++11 -pthread (project->properties>c++ build>settings> GCC C++ Compiler > command)
Установить опцию -pthread (project>properties>c++ build>settings> GCC C++ Linker)

Компилятор g++
Опция -g – отладка
g++ --std=c++11 -pthread –g code.cpp –o code.out
./code.out
gdb code.out
b main.cpp:10, r, info threads, thread 5…

Первая программа с потоками (C++11)

```
// compile with
// g++ -std=c++11 -pthread
file_name.cpp

#include <iostream>
#include <thread>

void function_to_call(int _id)
{
	std::cout << "Launched by
thread " << _id << std::endl;
}

int main()
{
	const int num_threads = 10;
	std::thread
threads[num_threads];
	for (int i = 0; i <
num_threads; ++i)
	{
		threads[i] =
std::thread(function_to_call,
i);
	}

	for (int i = 0; i < num_threads;
++i)
	{
		threads[i].join();
	}
	return 0;
}
```

Гонка потоков!

Где искать дополнительную информацию

Обучающие примеры
https://solarianprogrammer.com/2011/12/16/cpp-11-thread-tutorial/
https://solarianprogrammer.com/2012/02/27/cpp-11-thread-tutorial-part-2/
https://solarianprogrammer.com/2012/05/09/cpp-11-thread-tutorial-part-3/
http://www.quizful.net/post/multithreading-cpp11

Справочник по стандарту
http://en.cppreference.com/w/cpp/thread