

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ





Нижегородский государственный университет им. Н.И. Лобачевского
Институт информационных технологий, математики и механики

Метрики производительности, методика их сбора и анализа

Волокитин В.Д., Мееров И.Б.
Кафедра МОСТ

Школа ННГУ-Intel по оптимизации алгоритмов компьютерного зрения
2-8 февраля 2022 г.

Содержание

- ❑ Производительность аппаратных средств
- ❑ Производительность программного обеспечения
- ❑ Анализ производительности
- ❑ Roofline-модель

1. ПРОИЗВОДИТЕЛЬНОСТЬ АППАРАТНЫХ СРЕДСТВ

1. Производительность аппаратных средств

1.1. Основные метрики

- ❑ **Тактовая частота (GHz)** – скорость работы аппаратного средства (CPU, памяти)
- ❑ **Вычислительная скорость (GFLOPs)** – количество операций выполняемых за секунду
 - Целочисленные, с плавающей запятой (single precision, double precision)
- ❑ **Пропускная способность памяти (GB/s)** – объем передаваемых данных за секунду
 - Есть ли разница между чтением и записью?
 - NUMA-архитектура
- ❑ **Мощность (Watt)** – количество потребляемой энергии
- ❑ **Производные метрики:**
 - Flop/Byte, Flop/Watt ...

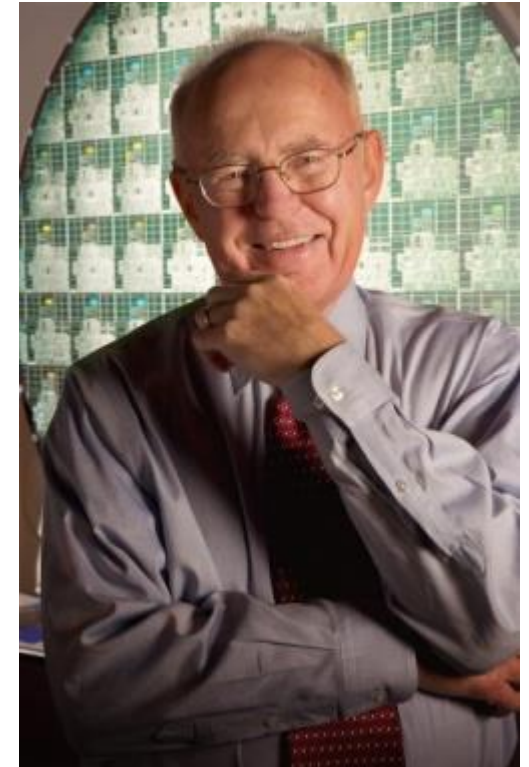
1. Производительность аппаратных средств

1.2. Закон Мура

□ Закон Мура:

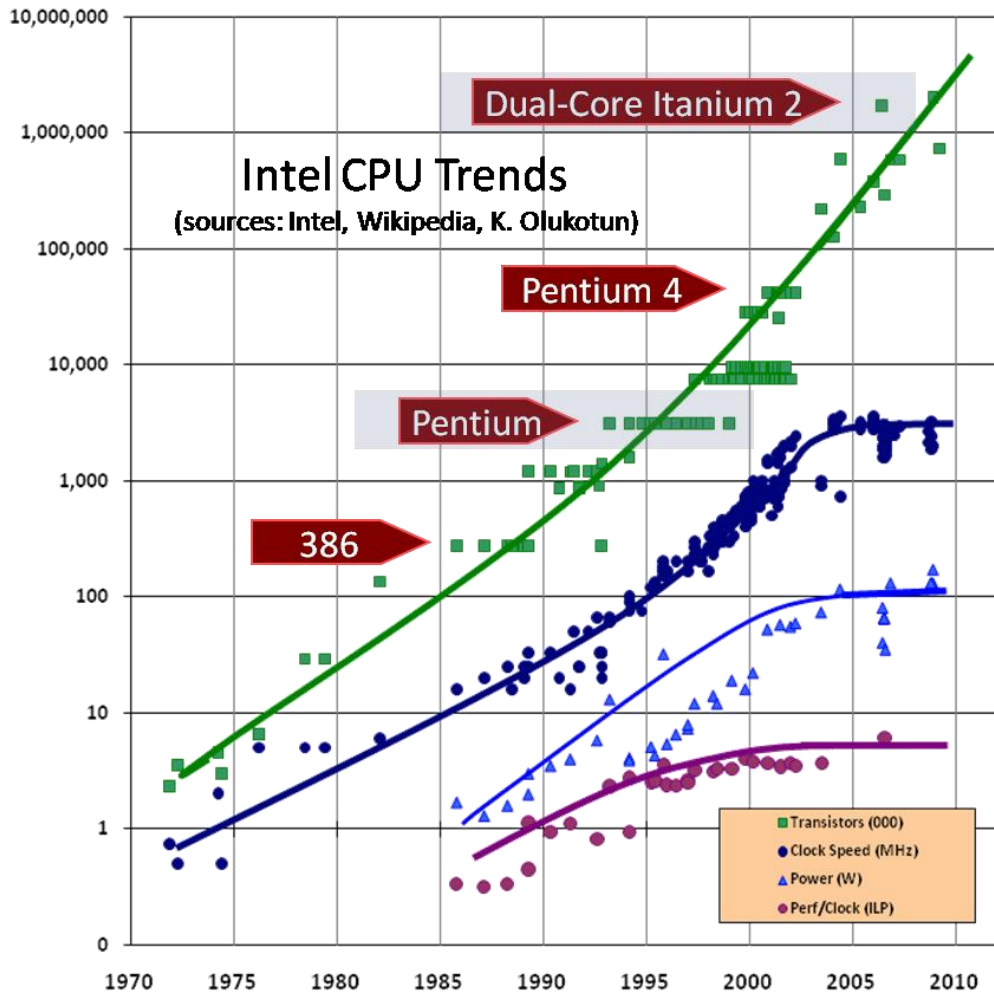
- Количество транзисторов, размещаемых на кристалле интегральной схемы, удваивается каждые 18 месяцев (1965 год)
- В 1975 Гордон Мур изменил срок удвоения на 24 месяца

□ Что это означает на самом деле?



1. Производительность аппаратных средств

1.3. Эволюция процессоров



- ❑ До сих пор закон Мура правдив, но он распространяется только на транзисторы
- ❑ Закон Мура не распространяется на:
 - Скорость памяти
 - Мощность
 - Количество инструкций на один такт (ILP)

Источник:

<http://www.gotw.ca/publications/concurrency-ddj.htm>

1. Производительность аппаратных средств

1.4. Теоретический пик производительности

- Пиковая производительность вычислений:

$$P * C * Vec * ILM * Clock,$$

где:

P – количество процессоров;

C – количество ядер на процессоре;

Vec – длина векторного регистра;

ILM – количество инструкций на один такт;

$Clock$ – частота процессора.

- Пиковая пропускная способность памяти:

$$MFreq * BPC * MWidht,$$

где:

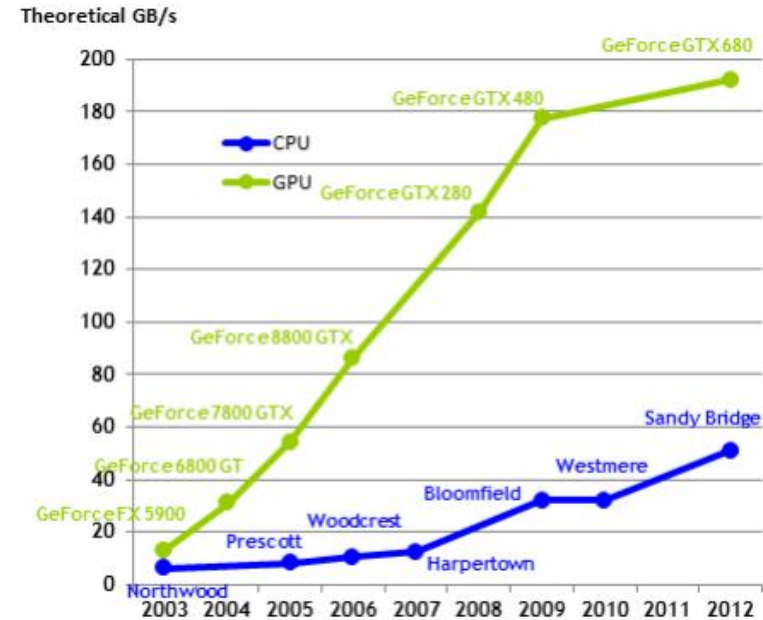
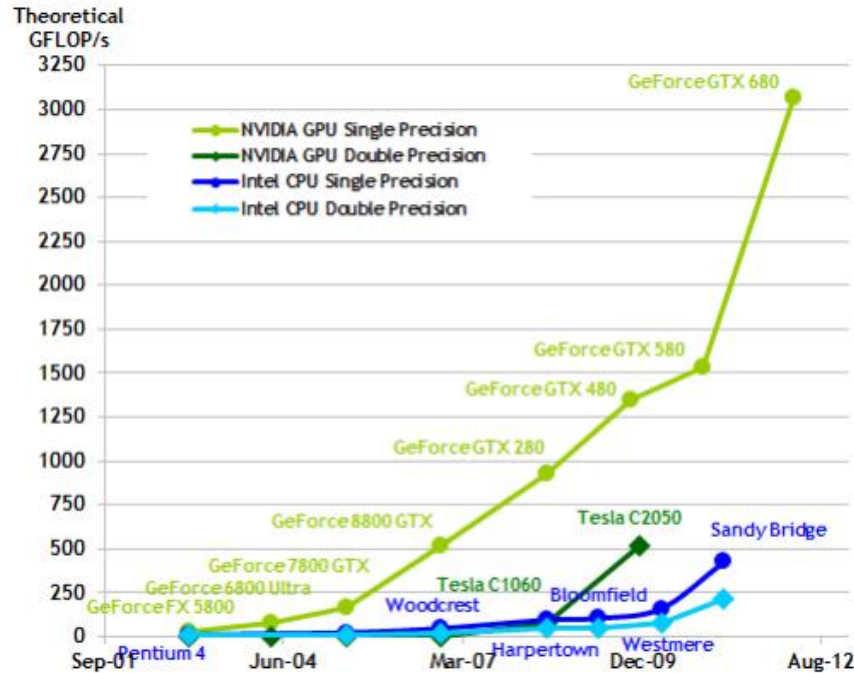
$MFreq$ – частота памяти (не путать с частотой процессора);

BPC – количество каналов памяти;

$MWidht$ – ширина шины.

1. Производительность аппаратных средств

1.5. GPU vs. CPU



❑ Все ли так хорошо на GPU?

Источник: Nvidia Cuda C Programming Guide version 4.2

1. Производительность аппаратных средств

1.6. Реальная производительность

- ❑ Пиковая производительность достигается:
 - При 100% задействовании вычислительных блоков
 - При 100% параллелизме
 - При передаче данных с максимальной пропускной способностью (на всех уровнях)

- ❑ Какая производительность в реальности?
 - Ни одно приложение не может работать с пиковой производительностью

- ❑ Как оценить реальную производительность системы?
 - Бенчмарки (Linpack, HPCG – высокопроизводительная линейная алгебра, Graph – работа с графами, обход в ширину)

2. ПРОИЗВОДИТЕЛЬНОСТЬ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2. Производительность ПО

2.1. Метрики производительности ПО

- ❑ **Пользователи** (Насколько хорошо работает приложение?):
 - Время работы
 - Ускорение
 - Масштабируемость
- ❑ **Разработчики** (Можно ли сделать лучше?):
 - Вычислительная сложность
 - Близость к пиковой производительности
- ❑ **Заказчики** (На сколько хорошо используется моя инфраструктура и тратится мой бюджет?):
 - Использование ресурсов (вычислительных, денежных и т.д.)

2. Производительность ПО

2.2. Ускорение (1)

□ Закон Амдала:

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

где:

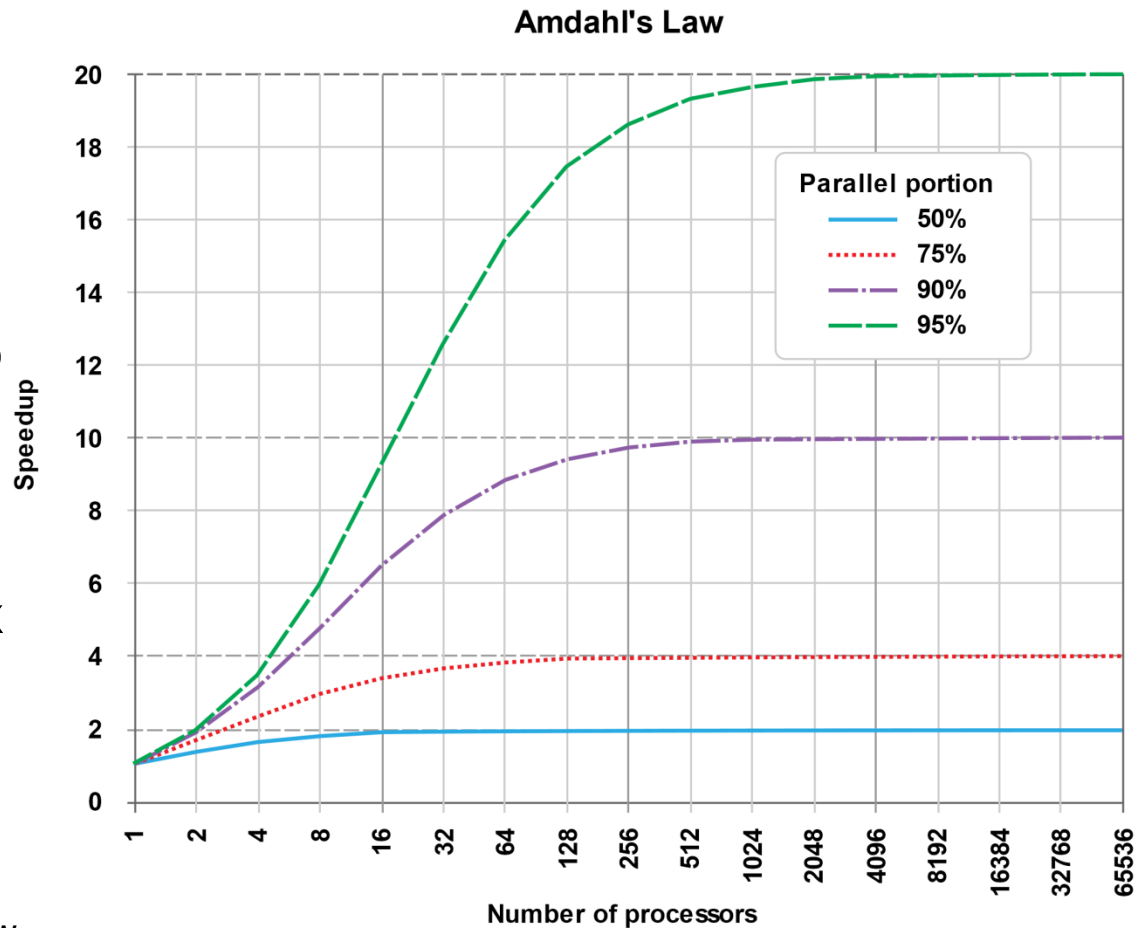
S_p - ускорение параллельного кода;

α – доля последовательного кода;

p – количество параллельных ресурсов.

Источник:

https://en.wikipedia.org/wiki/Amdahl%27s_law



2. Производительность ПО

2.2. Ускорение (2)

□ Ускорение на практике:

$$S = \frac{T_s}{T_p},$$

где:

T_s – время работы последовательной программы;

T_p – время работы параллельной программы.

□ Какое бывает ускорение:

- $S < p$ – сублинейное

- $S = p$ – линейное

Возможно ли $S > p$ – сверхлинейное?

2. Производительность ПО

2.3. Накладные расходы

- ❑ Межпроцессорные взаимодействия:

- Коммуникации
- Синхронизации

- ❑ Дисбаланс нагрузки:

- Неравномерное распределение задач



Простой в работе

- ❑ Дополнительные накладные расходы:

- Выделение памяти
- Распределение данных

- ❑ Как оценить накладные расходы?

2. Производительность ПО

2.3. Накладные расходы

- ❑ Межпроцессорные взаимодействия:

- Коммуникации
- Синхронизации

- ❑ Дисбаланс нагрузки:

- Неравномерное распределение задач



Простой в работе

- ❑ Дополнительные накладные расходы:

- Выделение памяти
- Распределение данных

- ❑ Как оценить накладные расходы?

$$T_o = pT_p - T_s$$

2. Производительность ПО

2.4. Эффективность

- **Эффективность** – соотношение между результатами и использованными ресурсами:

$$E = \frac{S}{p} = \frac{T_s}{pT_p}$$

Показывает насколько хорошо параллельная реализация использует параллелизм машины

- $E \leq 1$

$$E = \frac{1}{1 + \frac{T_o}{T_s}}$$

- T_s для параллельной реализации неизменно
- Увеличение $p \Rightarrow$ увеличение накладных расходов \Rightarrow снижение эффективности

2. Производительность ПО

2.5. Масштабируемость

- ❑ **Масштабируемость** – показывает, как изменится время работы приложения при изменении объема задачи и доступных ресурсов
- ❑ **Сильная масштабируемость** – показывает зависимость времени решения задачи от числа доступных ресурсов при фиксированном размере задачи
- ❑ **Слабая масштабируемость** – показывает зависимость времени решения задачи от числа доступных ресурсов при фиксированном размере задачи на единицу ресурса

3. АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ

3. Анализ производительности

3.1. Применение

- ❑ Ограничения при анализе производительности:
 - Замер производительности соответствует уникальному набору:
 - Приложение
 - Вычислительная машина
 - Набор данных
- ❑ Во время разработки кода:
 - Понимание узких мест приложений
 - Позволяет узнать, как улучшить свое приложение
 - Позволяет понять, когда остановиться в оптимизации кода
- ❑ Для прогнозирования:
 - Предсказать поведение приложения:
 - Для разных конфигураций вычислительных машин
 - Для разных применений

3. Анализ производительности

3.2. Относительные метрики

- ❑ Метрики:
 - Время работы
 - Ускорение
 - Эффективность
 - Масштабируемость
- ❑ Относительная производительность !!!
- ❑ Позволяют сравнивать только один и тот же алгоритм
- ❑ Не учитывают применение программного обеспечения в иных задачах

3. Анализ производительности

3.3. Оценка производительности

- ❑ Вычислительная производительность:
 - Достигнутая: $FLOPs = FLOP/T$
 - Эффективность: $E_s = FLOPs/Peak_FLOPs$
- ❑ Пропускная способность
 - Достигнутая: $BW = (READ + WRITE)/T$
 - Эффективность: $E_{bw} = BW/Peak_BW$
- ❑ Достигнутая производительность позволяет сравнивать разные алгоритмы
- ❑ Эффективность использования ресурсов позволяет учитывать характеристики вычислительной машины

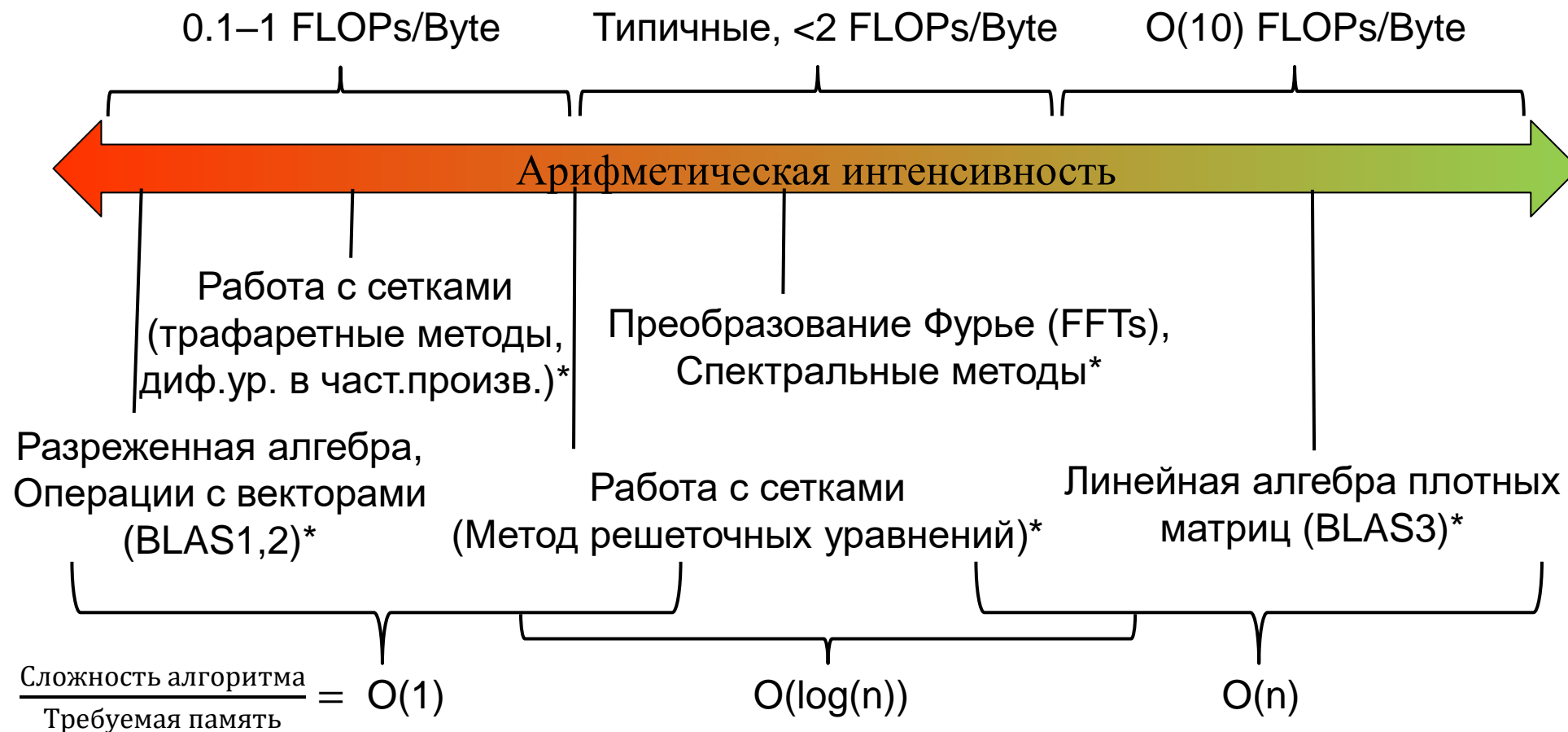
3. Анализ производительности

3.4. Арифметическая интенсивность (1)

- ❑ Арифметическая интенсивность – количество выполненных операций (обычно с плавающей запятой) на байт переданных данных
- ❑ Характеристика приложения, а не алгоритма
- ❑ Содержит только полезные операции. Не учитывает накладные расходы
- ❑ По арифметической интенсивности видно узкое место приложения:
 - Вычисления
 - Необходимо добавить вычислительных мощностей
 - Память
 - Необходимо увеличить пропускную способность памяти

3. Анализ производительности

3.4. Арифметическая интенсивность (2)



*Приблизительная арифметическая интенсивность при эффективной реализации алгоритма

4. ROOFLINE-МОДЕЛЬ

4. Roofline-модель

4.1. Определение

- ❑ **Roofline-модель** – визуальная интуитивно понятная модель производительности
 - Учитывает характеристики приложения через арифметическую интенсивность
 - Учитывает платформу через технические характеристики оборудования
- ❑ Определяет границы производительности
- ❑ Дает понимания о возможных оптимизациях
- ❑ Roofline-модель – это график в осях арифметической или операционной интенсивности (FLOPs/Byte) и производительности (GFLOPs)

Williams S., Waterman A., Patterson D. Roofline: an insightful visual performance model for multicore architectures //Communications of the ACM. – 2009. – Т. 52. – №. 4. – С. 65-76.

4. Roofline-модель

4.2. Вычислительные пики

- Для примера рассмотрим платформу:

- 2xIntel Xeon Platinum 8260L (2x24 ядра, 2.4 GHz, AVX512)

- Пик скалярных вычислений:

$$\text{Scalar Peak} = 2(p) * 24(c) * 2(ILM) * 2.4GHz = 230.4 GFLOPs$$

- Пик векторных вычислений (двойная точность):

$$\text{Vector Peak} = 2 * 24 * 2 * 8(vec) * 2.4GHz = 1843.2 GFLOPs$$

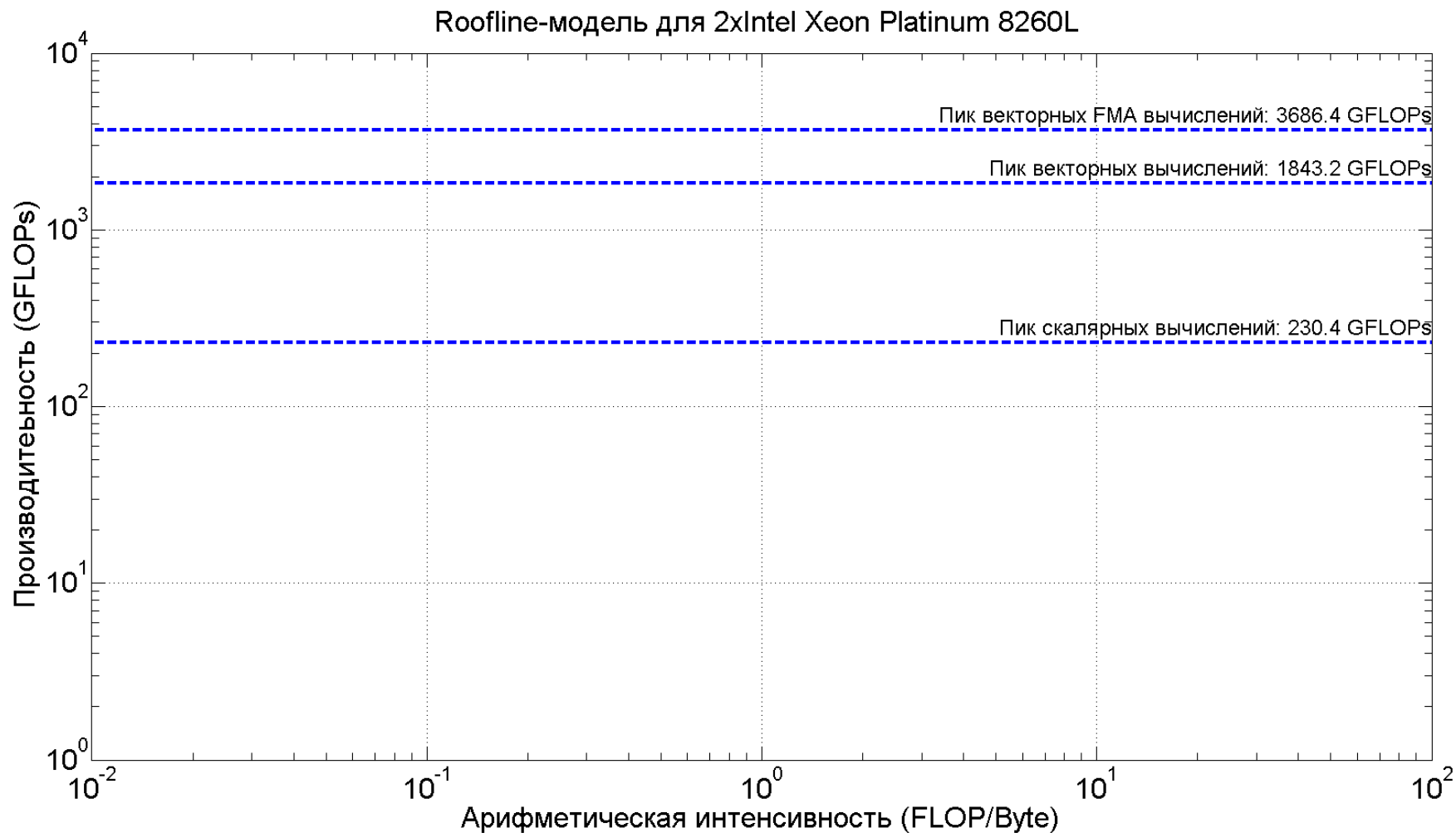
- Пик векторных вычислений с FMA (двойная точность):

$$\begin{aligned}\text{Vector FMA Peak} &= 2 * 24 * 4(ILM + FMA) * 8 * 2.4GHz \\ &= 3686.4 GFLOPs = 3.6 TFLOPs\end{aligned}$$

- FMA – вычисление $a * b + c$, которое может выполняться в современных архитектурах как одна операция без промежуточного округления

4. Roofline-модель

4.2. Вычислительные пики (2)



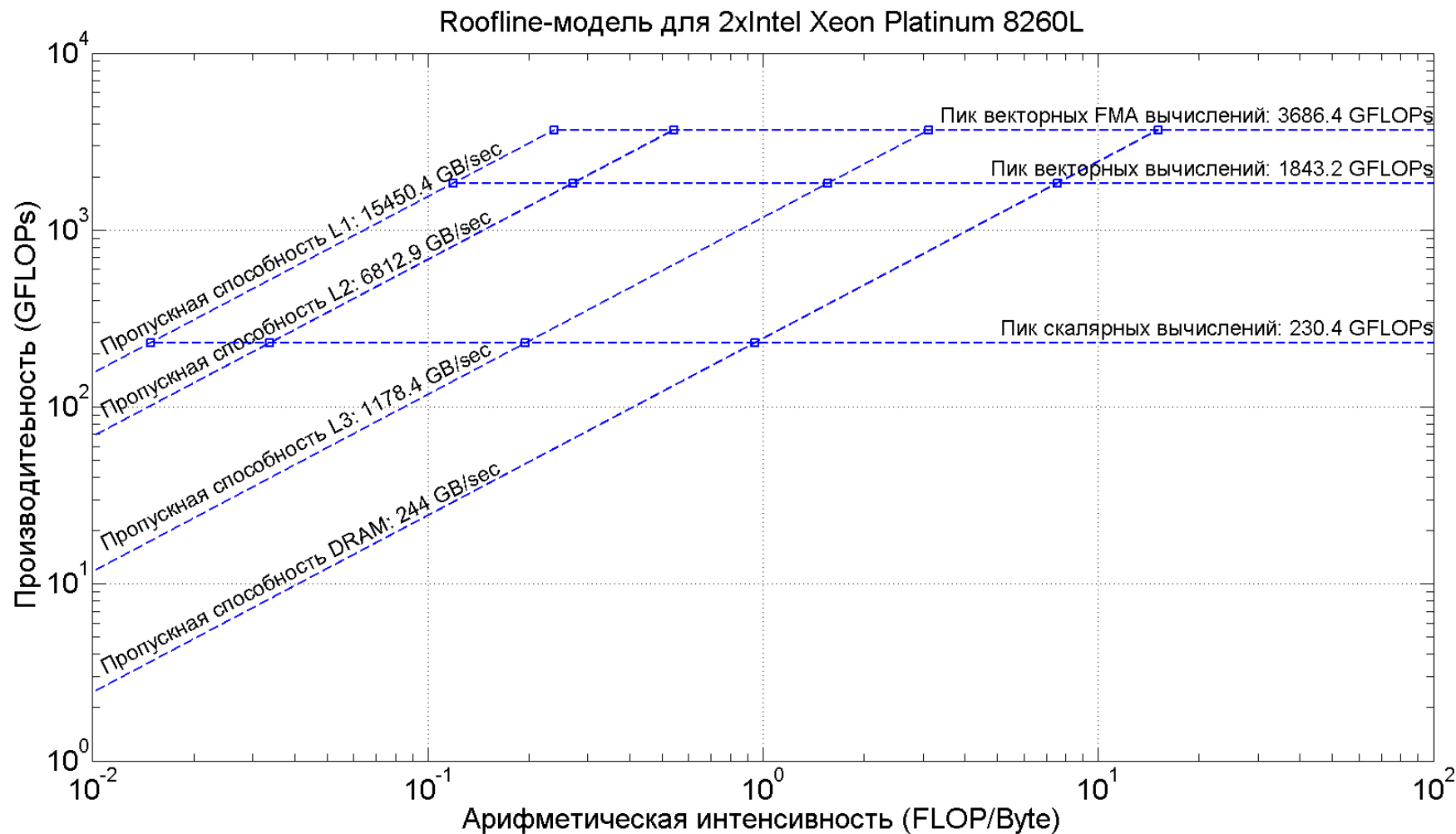
4. Roofline-модель

4.3. Пропускная способность памяти

- ❑ Теоретическую оценку пропускной способности памяти не используют для Roofline-модели, так как она может сильно отличаться от практических значений
- ❑ Используют значения, полученные с помощью бенчмарков
- ❑ Для рассматриваемой платформы:
 - Пропускная способность DRAM: 244 GB/sec
 - Пропускная способность кэша L3: 1178.4 GB/sec
 - Пропускная способность кэша L2: 6812.9 GB/sec
 - Пропускная способность кэша L1: 15450.4 GB/sec
- ❑ Как на графике отобразить пропускную способность памяти?

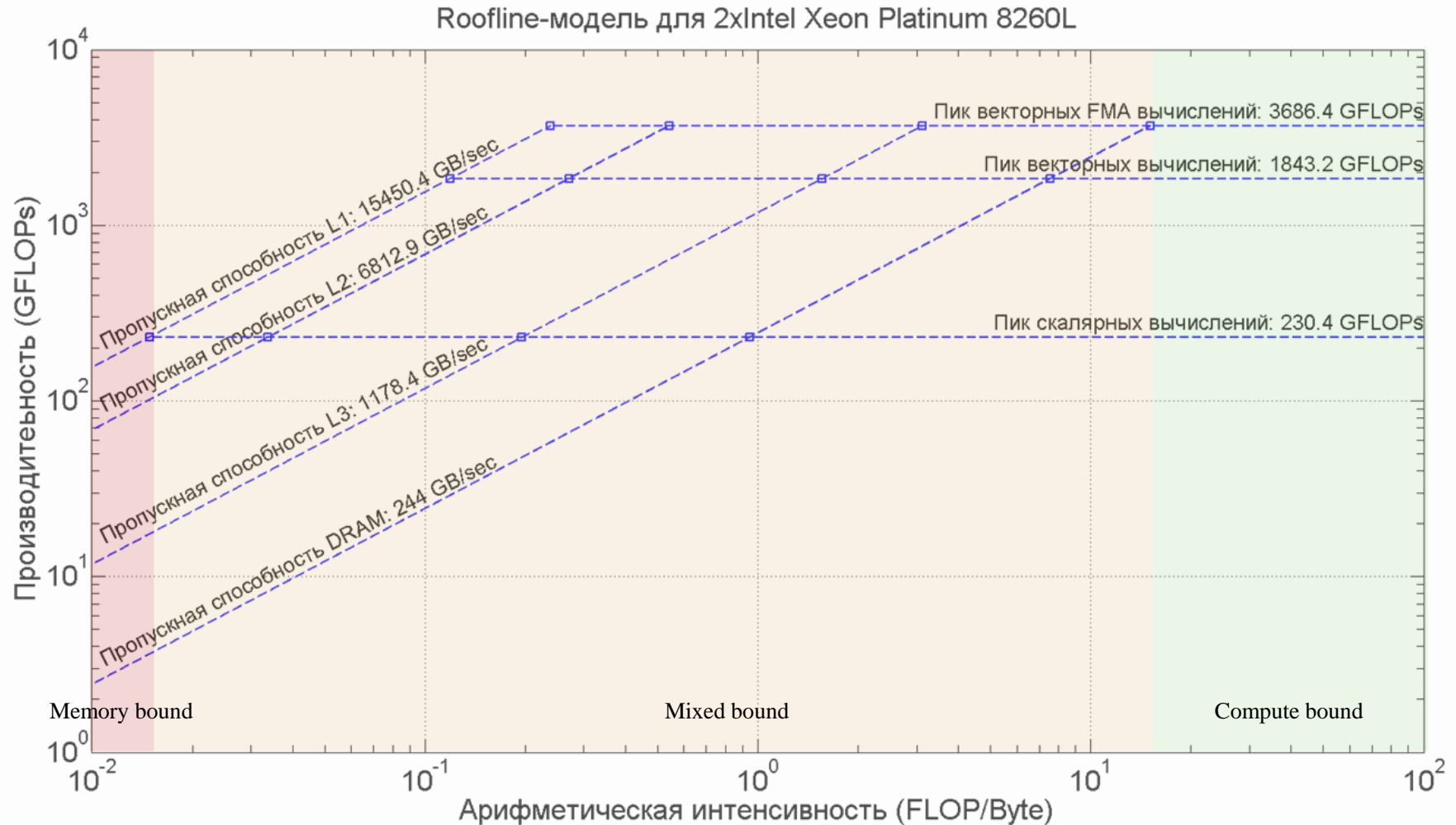
4. Roofline-модель

4.3. Пропускная способность памяти (2)



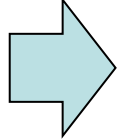
4. Roofline-модель

4.4. Узкие места программы



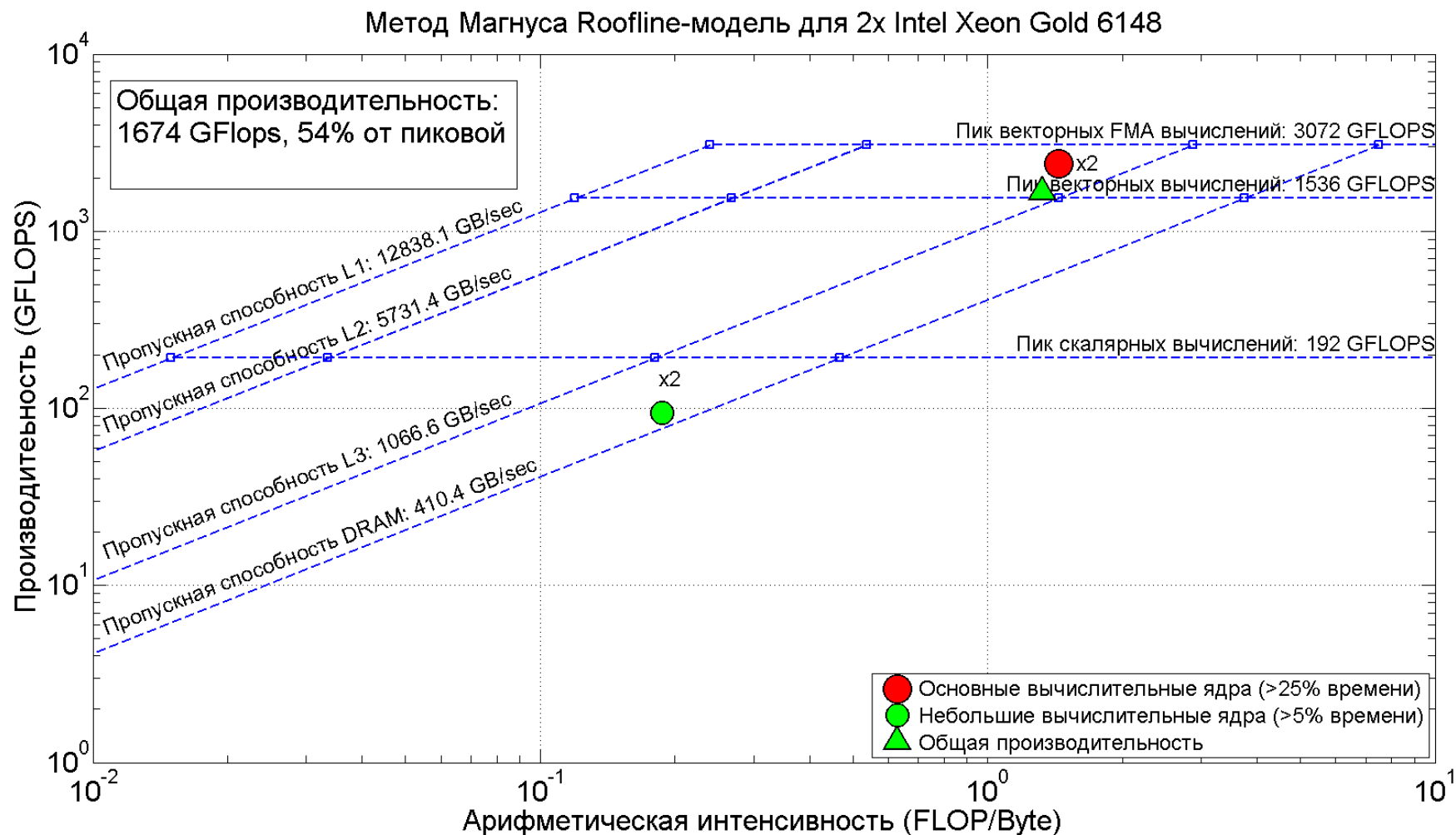
4. Roofline-модель

4.5. Циклы и функции приложения

- ❑ Каждый цикл в программе обладает следующими характеристиками:
 - Арифметическая интенсивность (зависит от скомпилированного кода)
 - Объем вычислений
 - Время выполнения
-  Производительность цикла
- ❑ Каждая функция в программе может быть представлена как совокупность своих циклов и функций
 - ❑ Все циклы и функции приложения могут быть отображены на графике Roofline-модели

4. Roofline-модель

4.6. Циклы и функции приложения. Пример



5. ЗАКЛЮЧЕНИЕ

Заключение

- ❑ Рассмотрены основные метрики производительности аппаратных средств:
 - Вычислительная скорость
 - Пропускная способность памяти
 - Мощность
- ❑ Рассмотрены основные метрики производительности программного обеспечения:
 - Ускорение и эффективность
 - Масштабируемость
 - Арифметическая интенсивность
- ❑ Рассмотрена Roofline-модель

Авторский коллектив курса

- ❑ Мееров Иосиф Борисович, к.т.н., доцент, зам. зав. каф. МОСТ ИИТММ
- ❑ Волокитин Валентин Дмитриевич, аспирант каф. МОСТ ИИТММ

Контакты

Нижегородский государственный университет

<http://www.unn.ru>

Институт информационных технологий, математики и механики

<http://www.itmm.unn.ru>

Мееров И.Б.

meerov@vmk.unn.ru