Новосибирский государственный университет
Факультет информационных технологий
Кафедра параллельных вычислений

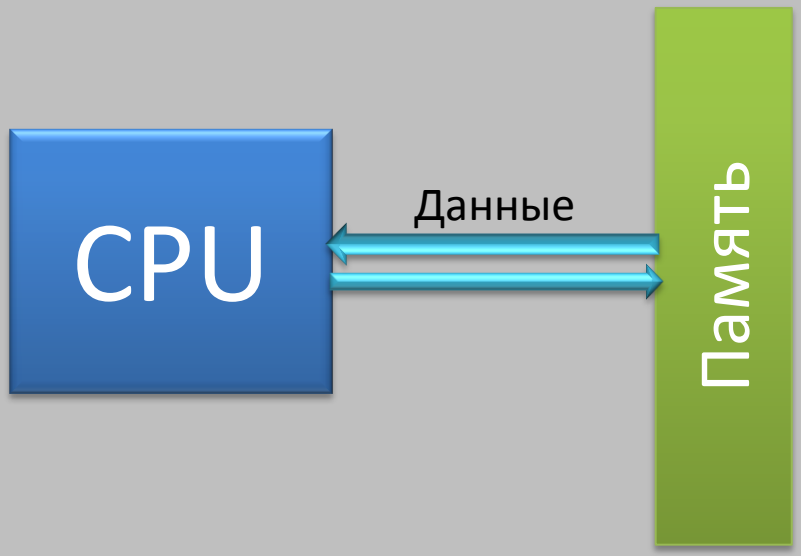
Эффективное программирование современных микропроцессоров и мультипроцессоров

Анализ производительности. Roofline Performance Model

Преподаватели:
Киреев С.Е.
Калгин К.В.

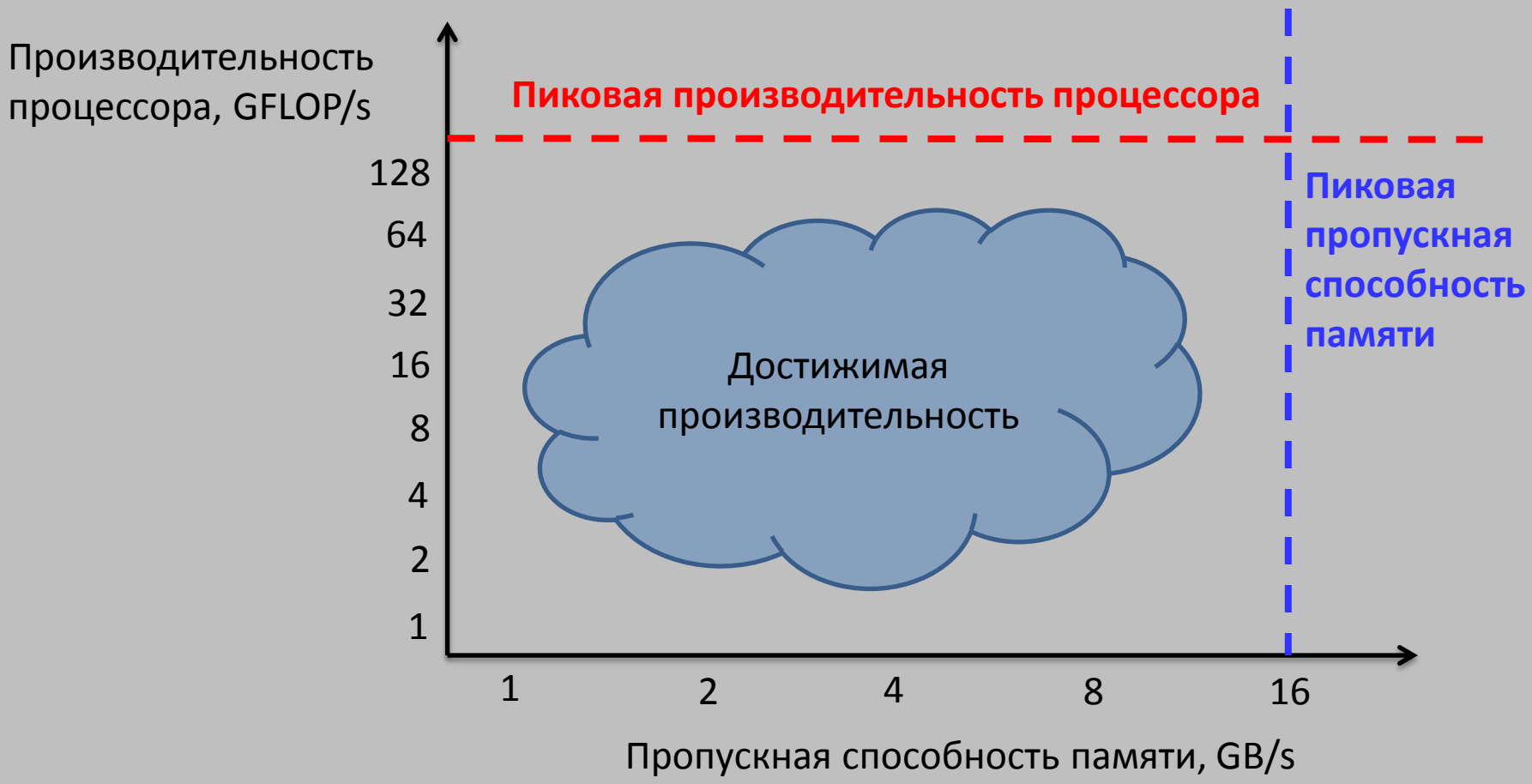
ИВМиМГ СО РАН
Лаборатория синтеза параллельных программ

Упрощённая модель вычислителя

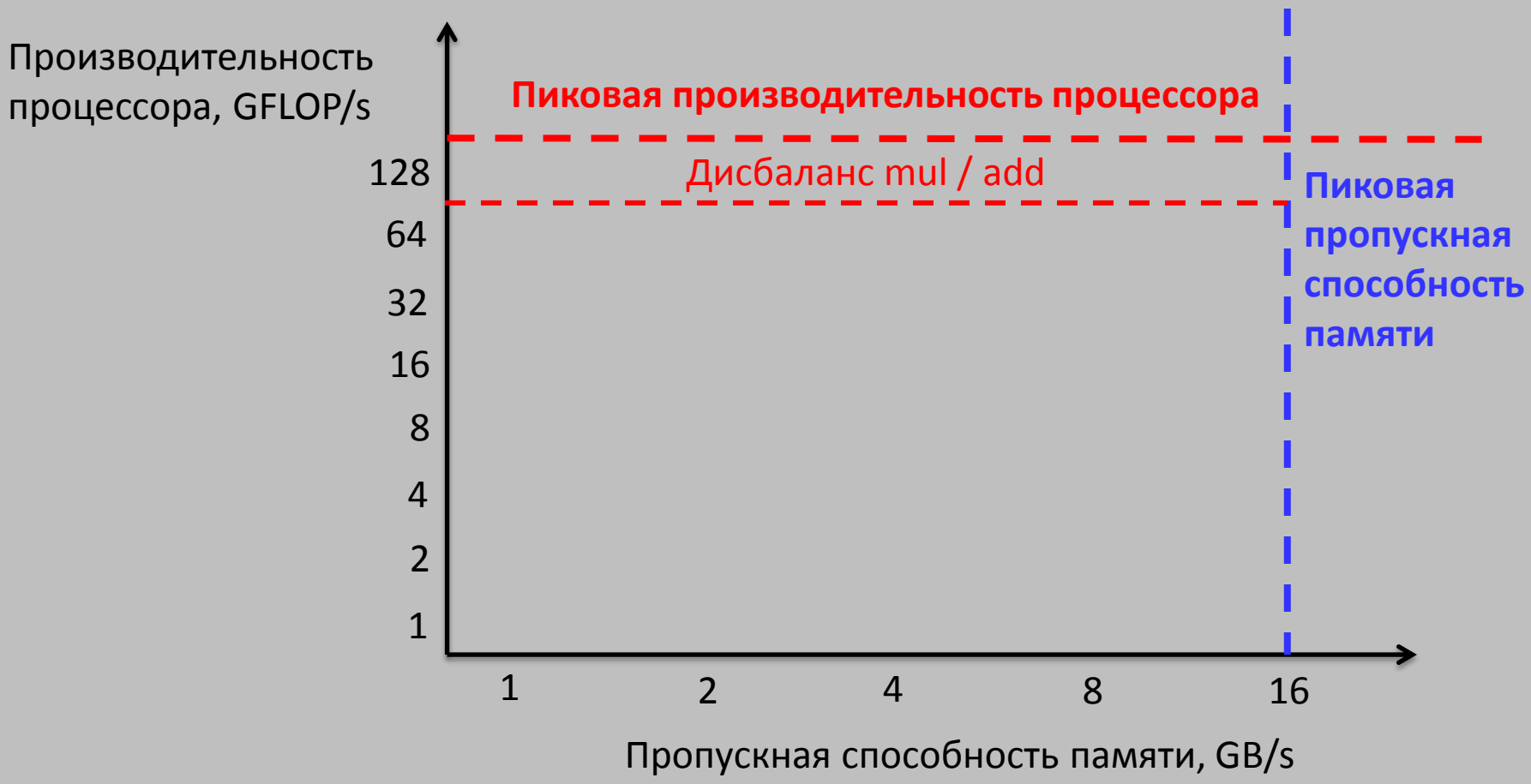


Характеристики производительности	Процессор	Память
Время	Время выполнения операции, ns	Время доступа к элементу данных, ns
Пропускная способность	Число операций в единицу времени, Op/s	Число элементов данных в единицу времени, B/s

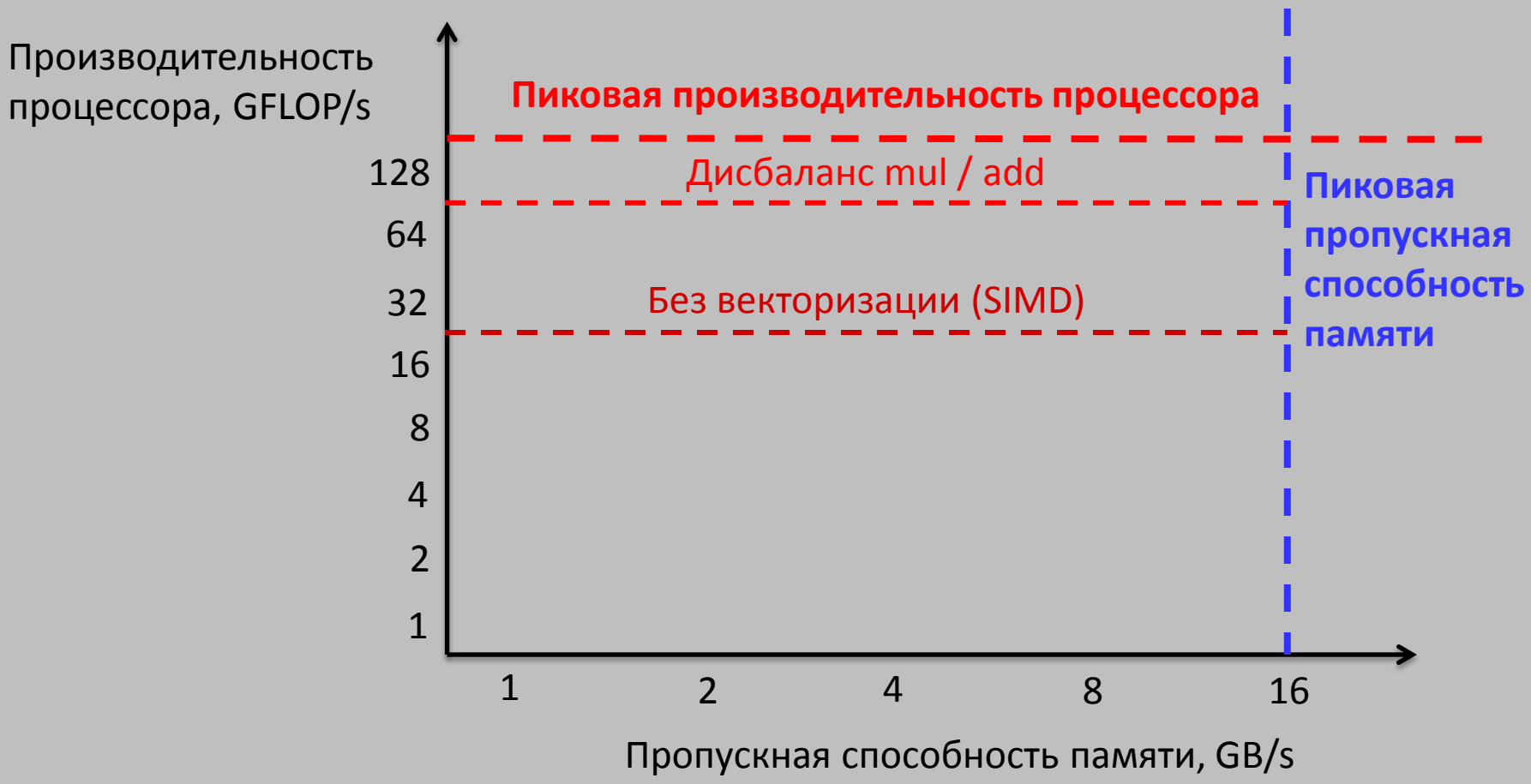
Границы производительности



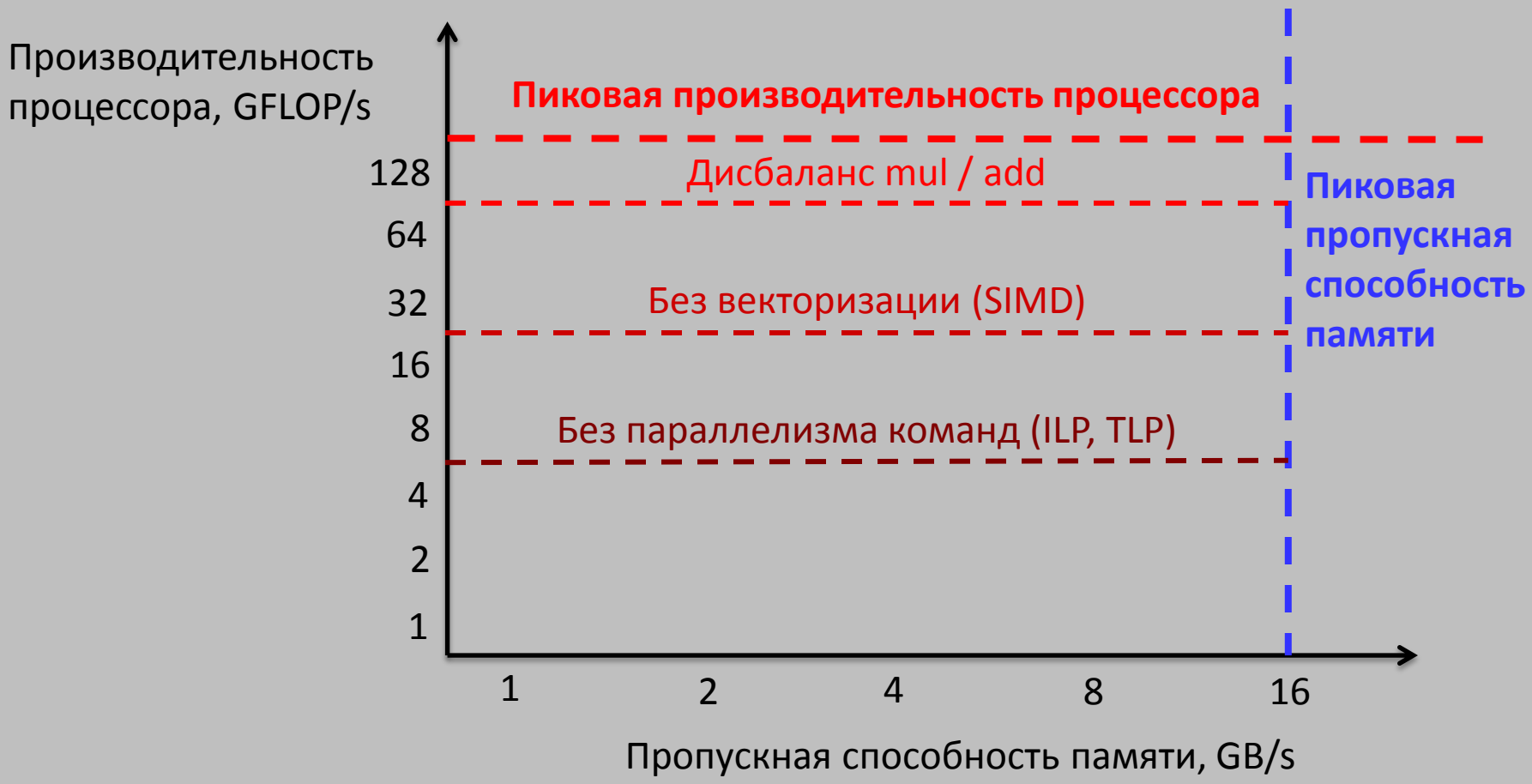
Границы производительности



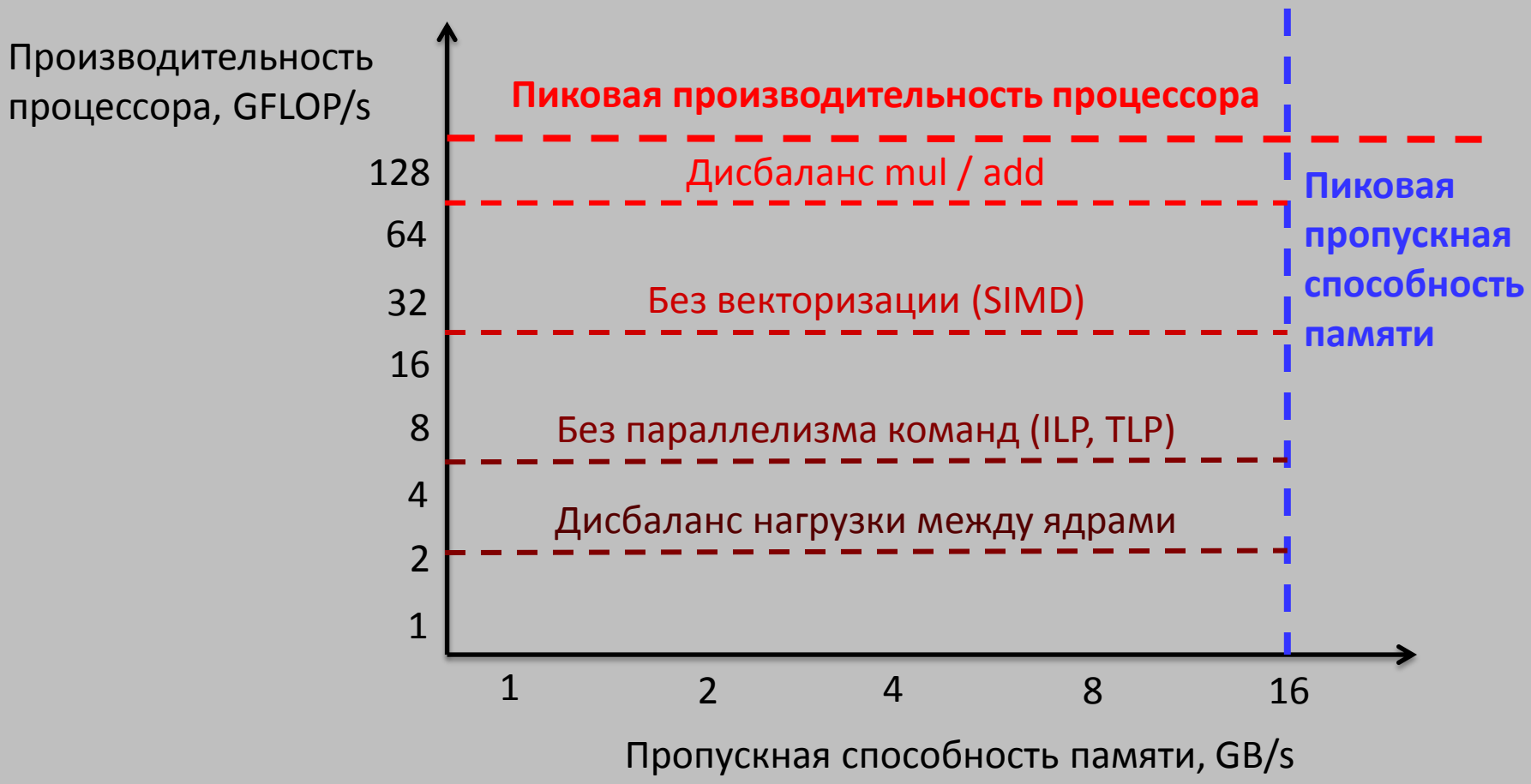
Границы производительности



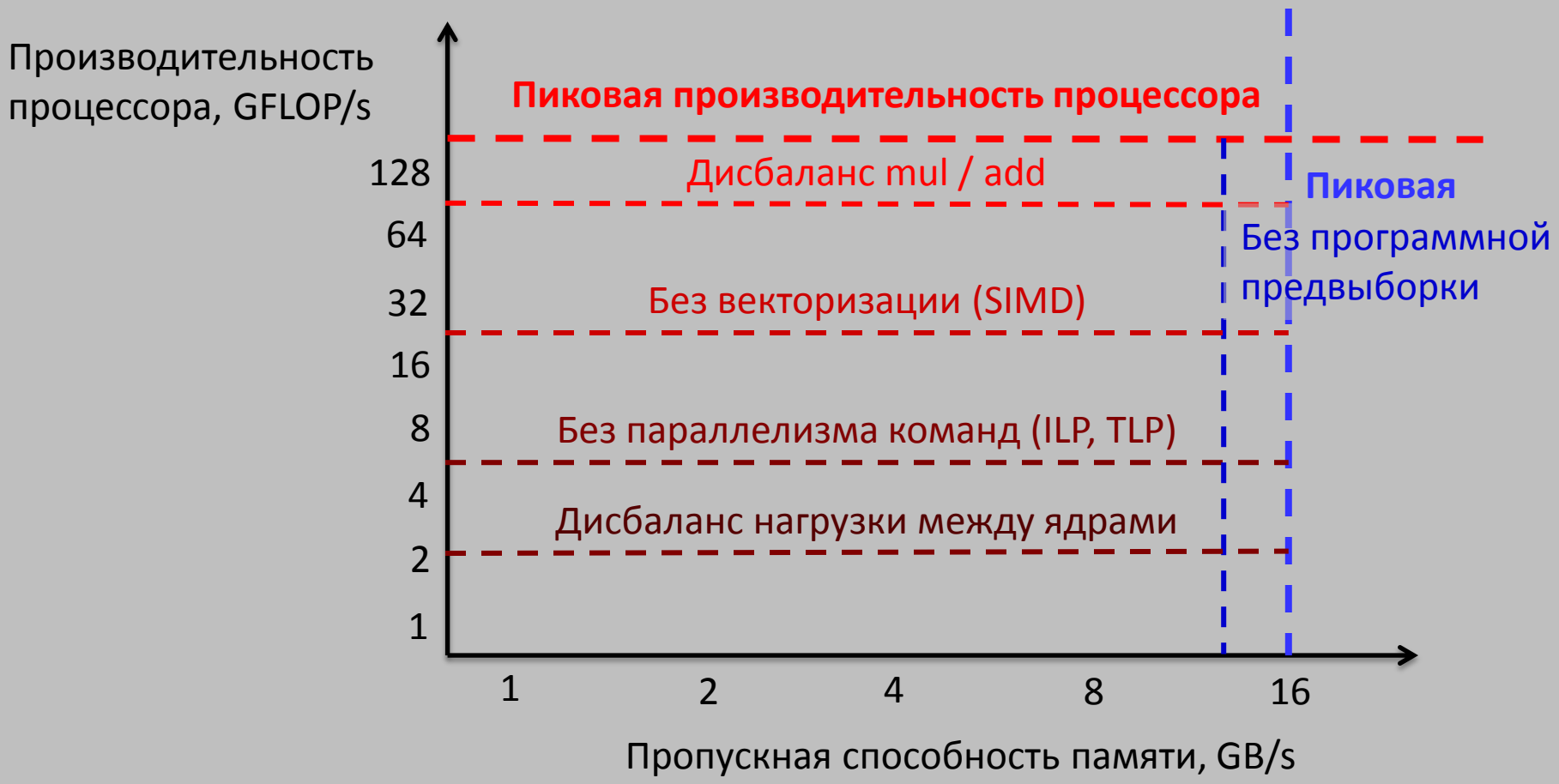
Границы производительности



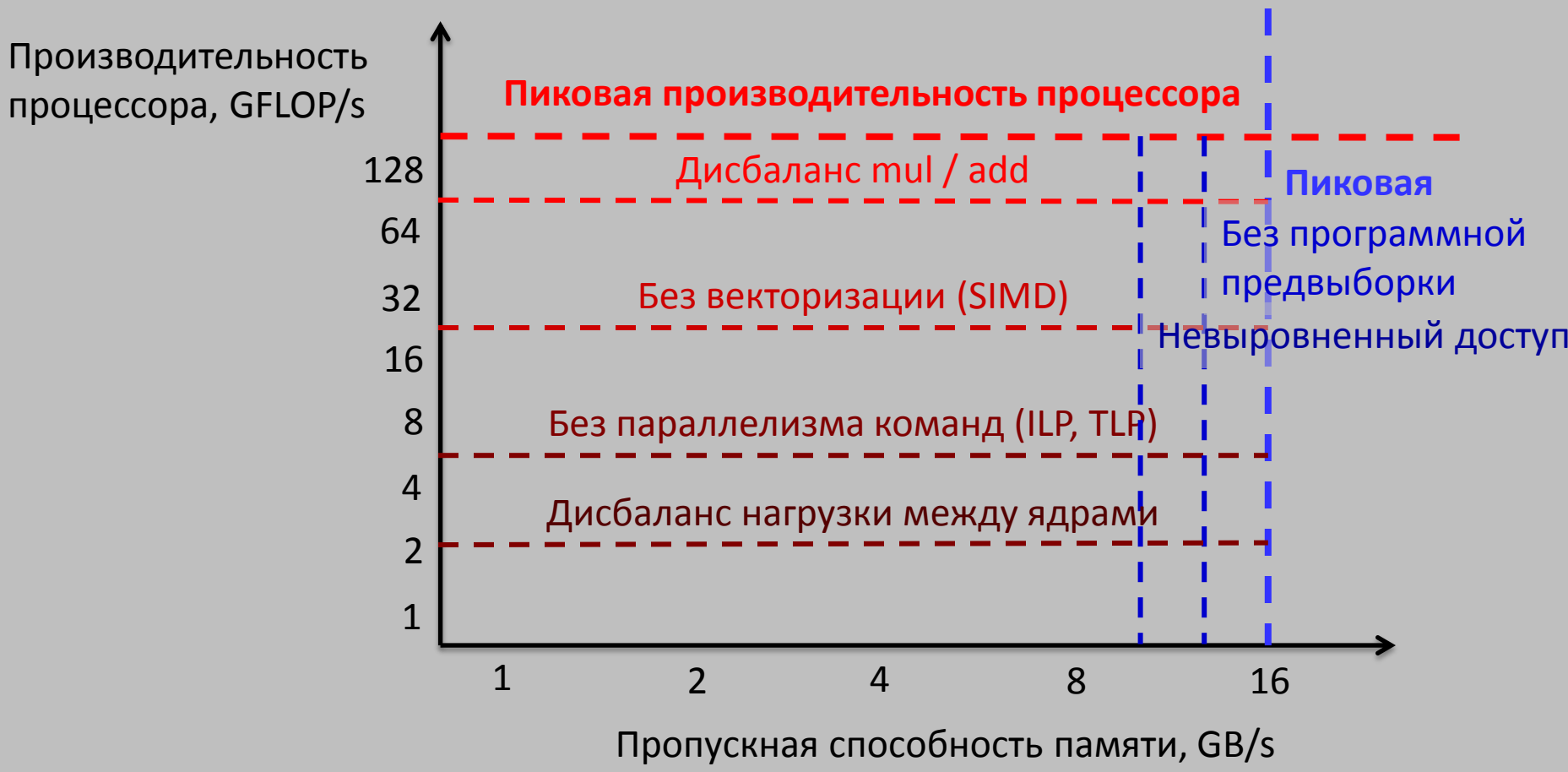
Границы производительности



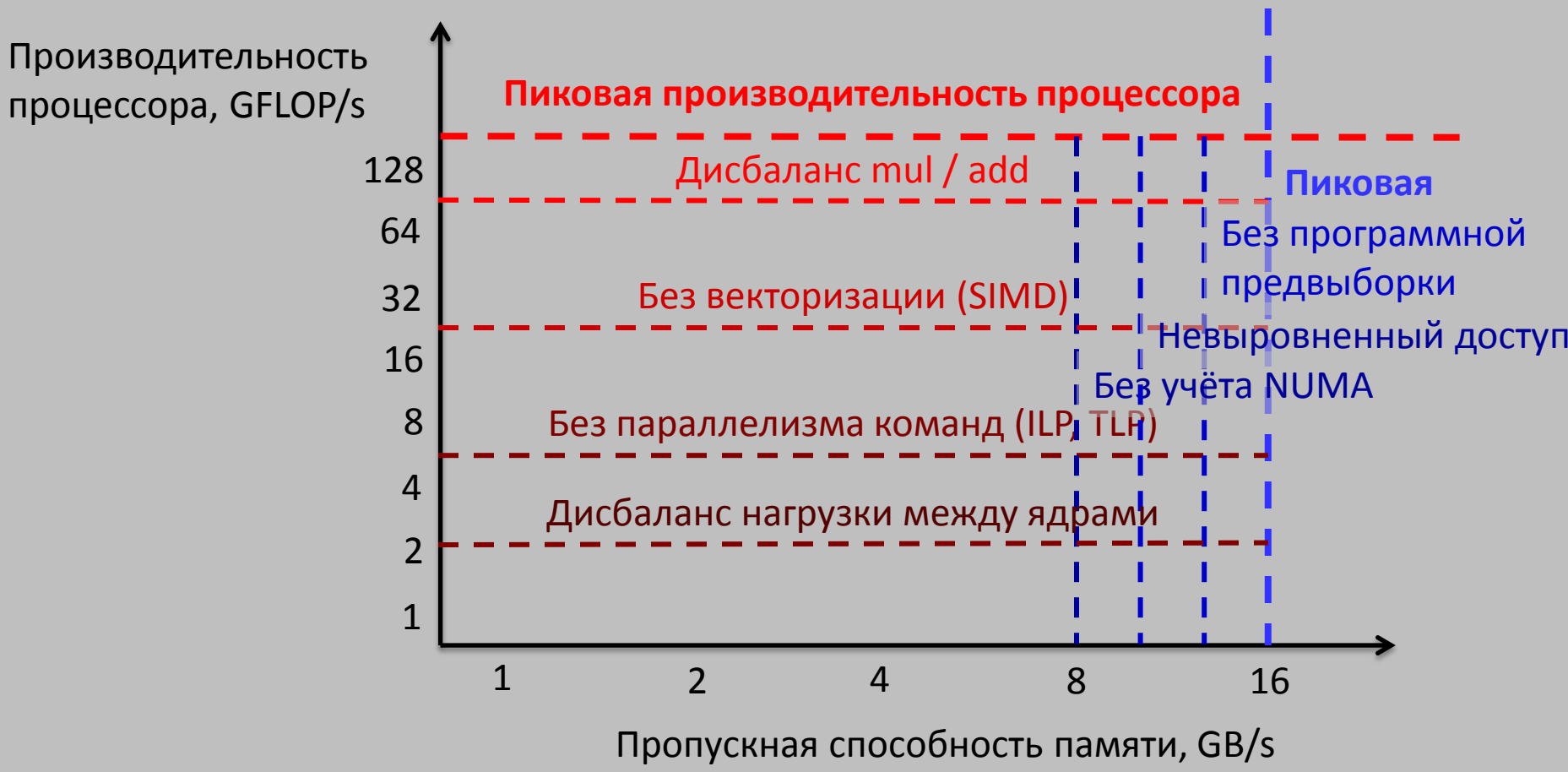
Границы производительности



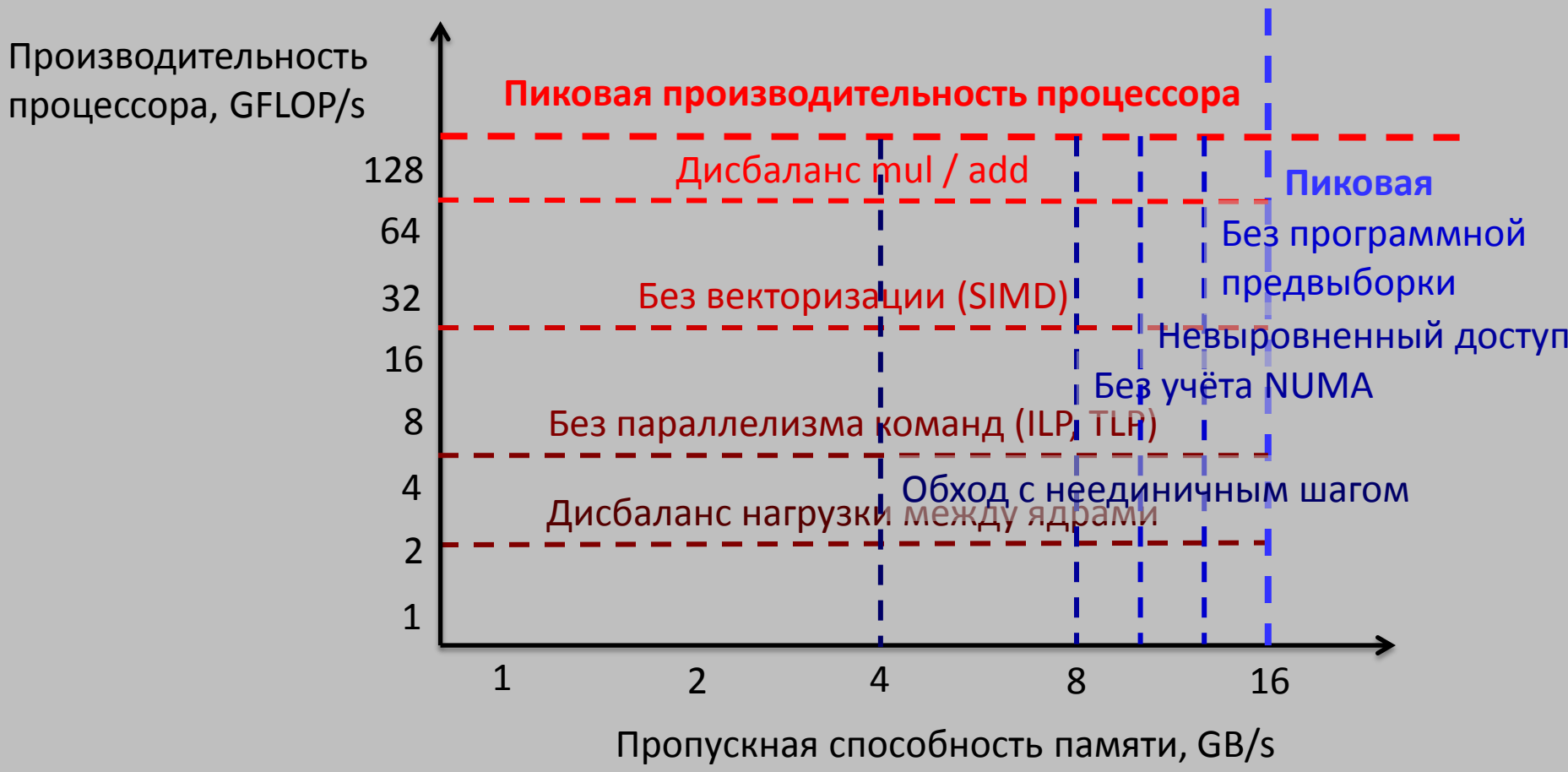
Границы производительности



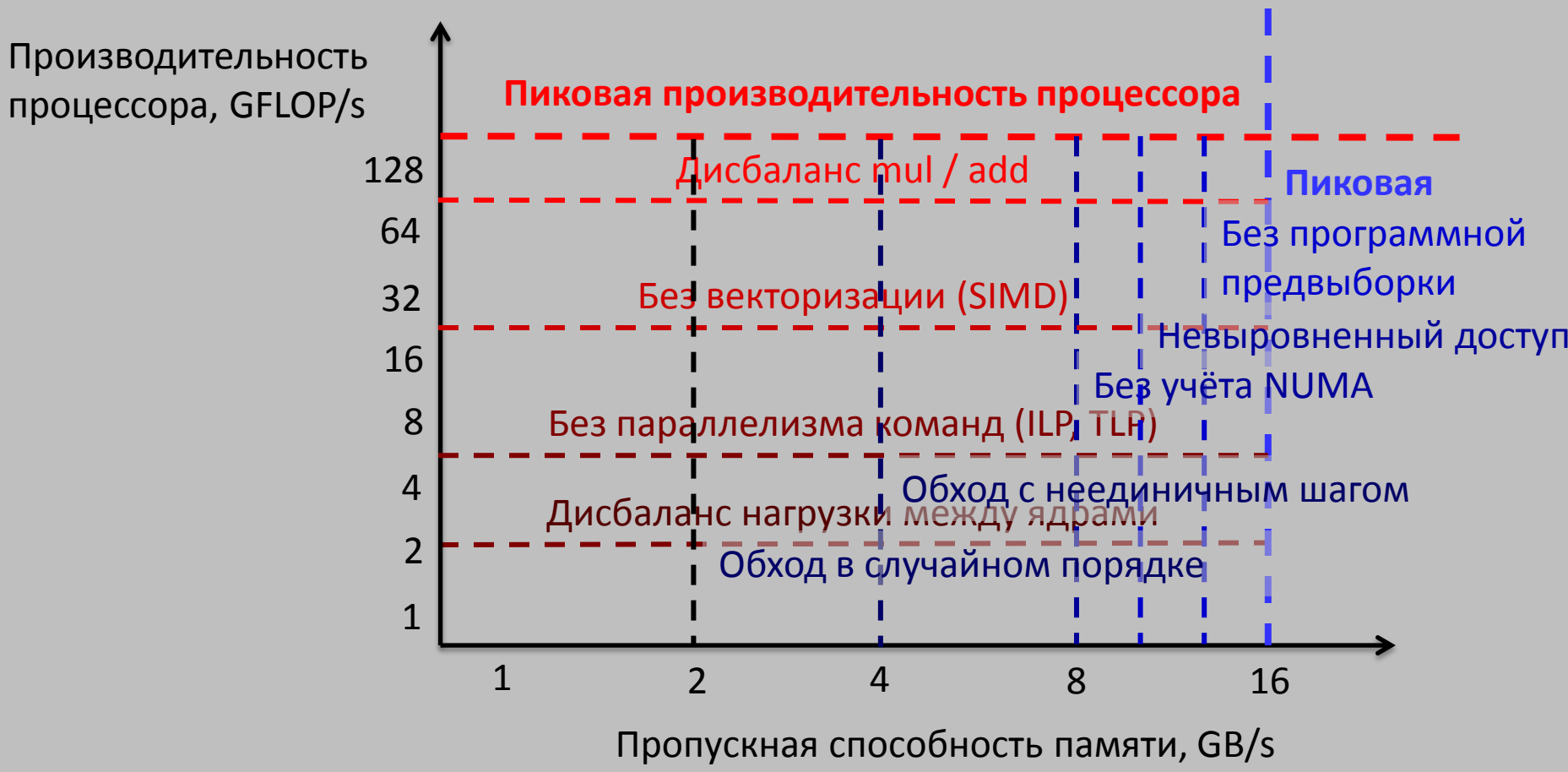
Границы производительности



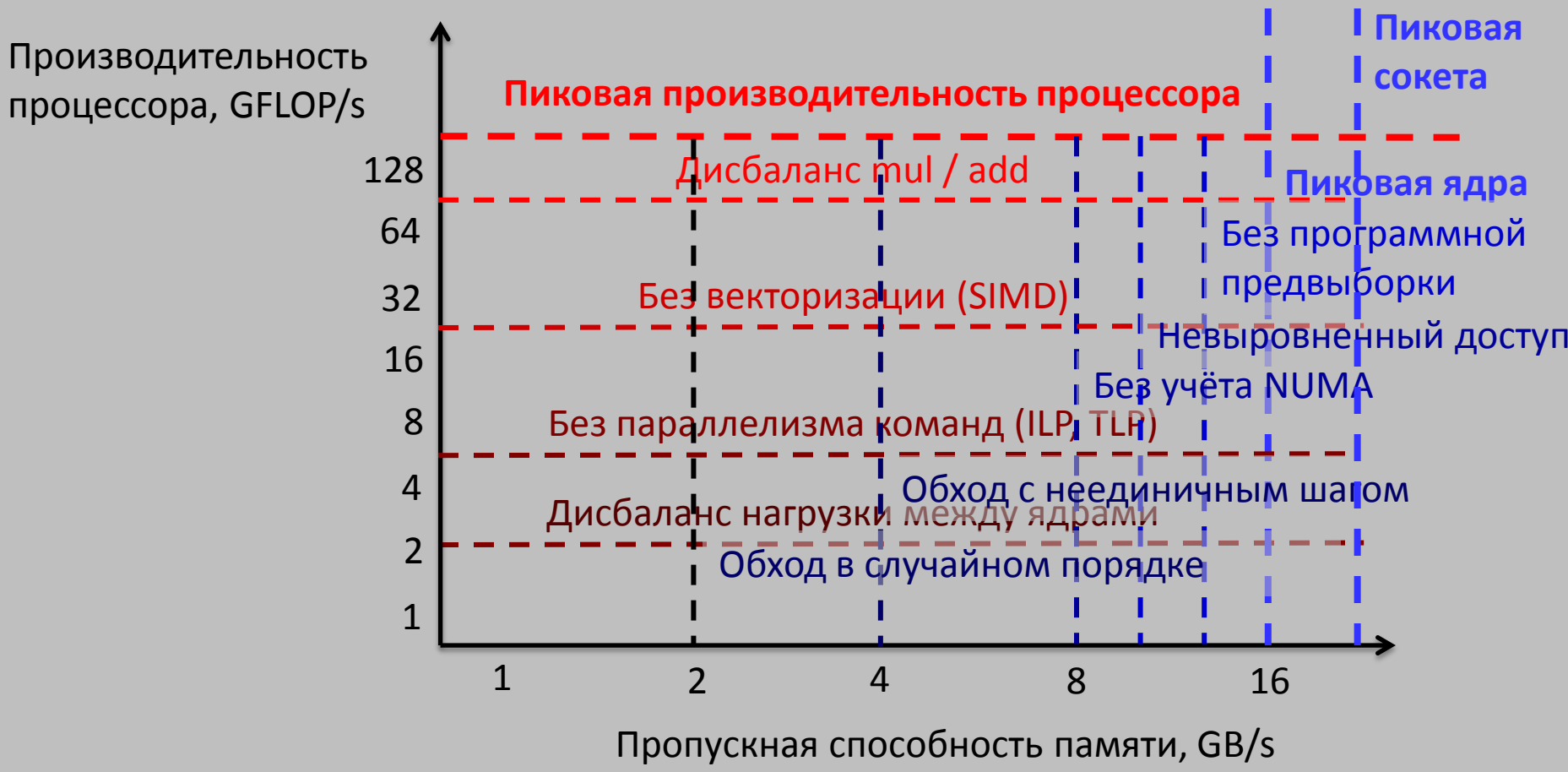
Границы производительности



Границы производительности

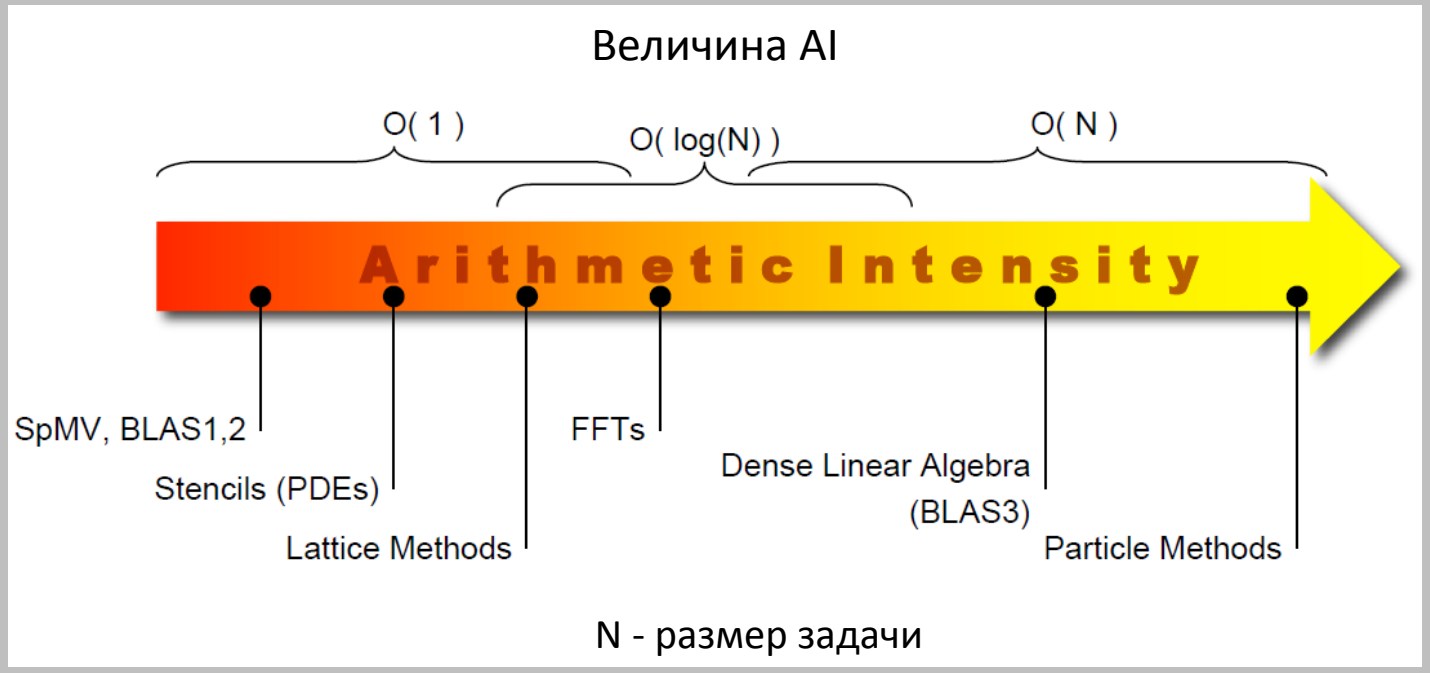


Границы производительности



Арифметическая интенсивность

- **Арифметическая интенсивность** (Arithmetic Intensity) – характеристика программы, определяющая отношение общего числа арифметических операций к общему числу передач данных из памяти / в память (FLOPS/B).



Арифметическая интенсивность

- Примеры:
  - Копирование вектора: x[] = y[]

```
for (int i=0;i<N;i++) x[i] = y[i];
```

    - AI = 0 FLOPS / (N * 2 * sizeof(double) B) = 0

  - Сложение векторов: x[] += y[]

```
for (int i=0;i<N;i++) x[i] += y[i];
```

    - AI = N FLOPS / (N * (2+1) * sizeof(double) B) = 1/24 ≈ 0.0417

Арифметическая интенсивность

- Примеры:
  - Копирование вектора: x[] = y[]
    - AI = 0
  - Сложение векторов: x[] += y[]
    - AI ≈ 0.0417
  - Умножение матрицы на вектор: y[] = a[][]*x[]

```
for (int i=0;i<N;i++) {
  double s = 0;
  for (int j=0;j<N;j++) s += a[i][j] * x[j];
  y[i] = s;
}
```

- AI = $(2*N^2)$ FLOPS / $((2*N^2 + 2*N)$ *sizeof(double) B) ≈ 2/16 = 0.125

- AI = $(2*N^2)$ FLOPS / $((N^2 + 3*N)$ *sizeof(double) B) ≈ 2/8 = 0.25

Арифметическая интенсивность

- Примеры:
  - Копирование вектора: x[] = y[]
    - AI = 0
  - Сложение векторов: x[] += y[]
    - AI ≈ 0.0417
  - Умножение матрицы на вектор: y[] = a[][]*x[]
    - AI = 0.125 AI = 0.25
  - Умножение матриц: c[][] = a[][] * b[][]

```
for (int i=0;i<N;i++)
  for (int j=0;j<N;j++) {
    double aij = a[i][j];
    for (int k=0;k<N;k++) c[i][k] += aij * b[j][k];
  }
```

- AI = (2*N^3 FLOPS) / ((3*N^3+N^2)*sizeof(double) B) ≈ 2/24 = 0.083
- AI = (2*N^3 FLOPS) / ((N^3 + 3*N^2)*sizeof(double) B) ≈ 2/8 = 0.25

Арифметическая интенсивность

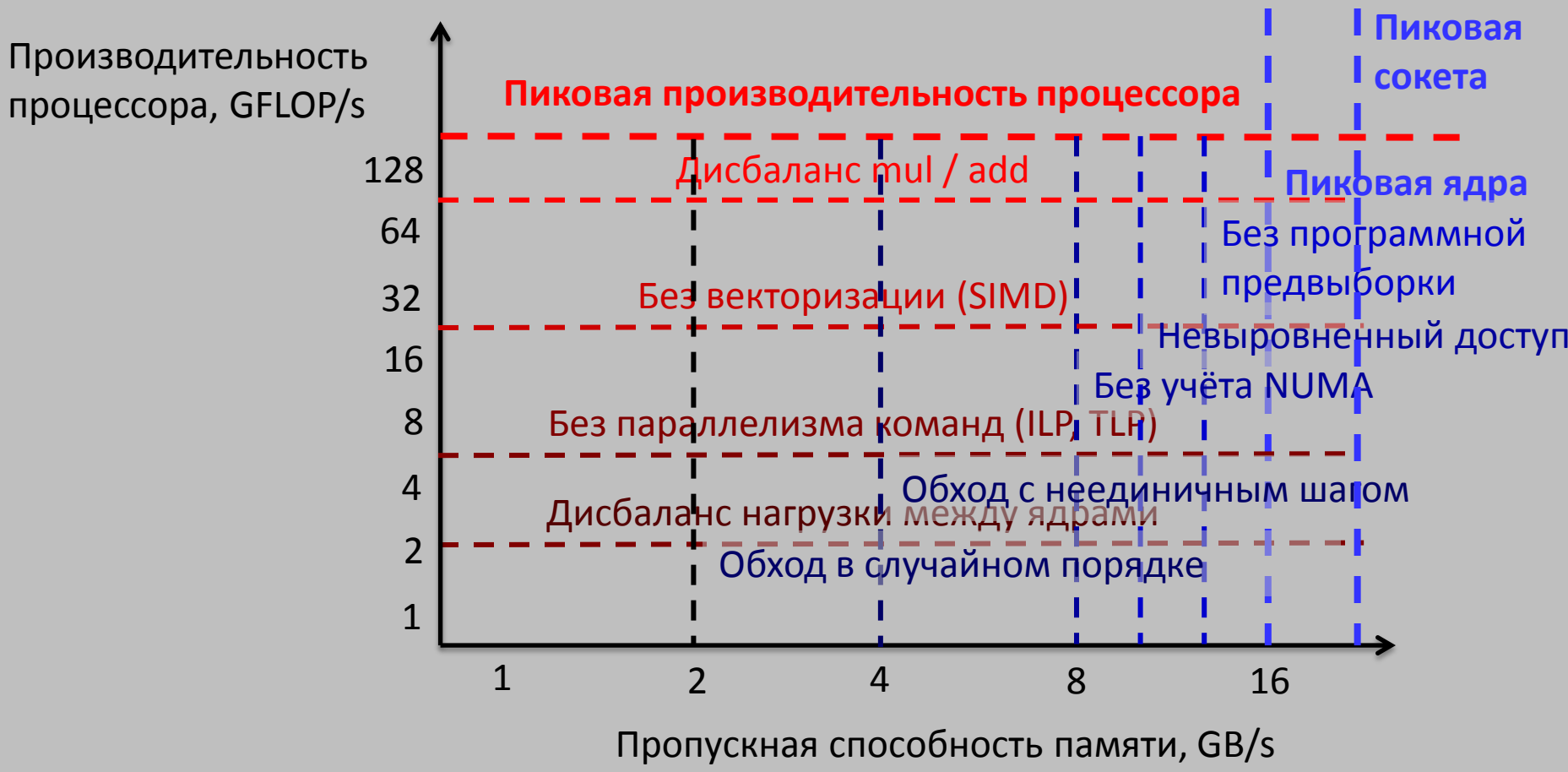
- Примеры:
  - Копирование вектора: x[] = y[]
    - AI = 0
  - Сложение векторов: x[] += y[]
    - AI ≈ 0.0417
  - Умножение матрицы на вектор: y[] = a[][]*x[]
    - AI = 0.125 AI = 0.25
  - Умножение матриц: c[][] = a[][] * b[][]
    - AI ≈ 0.083 AI = 0.25
  - Вычисление суммы ряда:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots + \frac{x^n}{n!} + \ldots$$

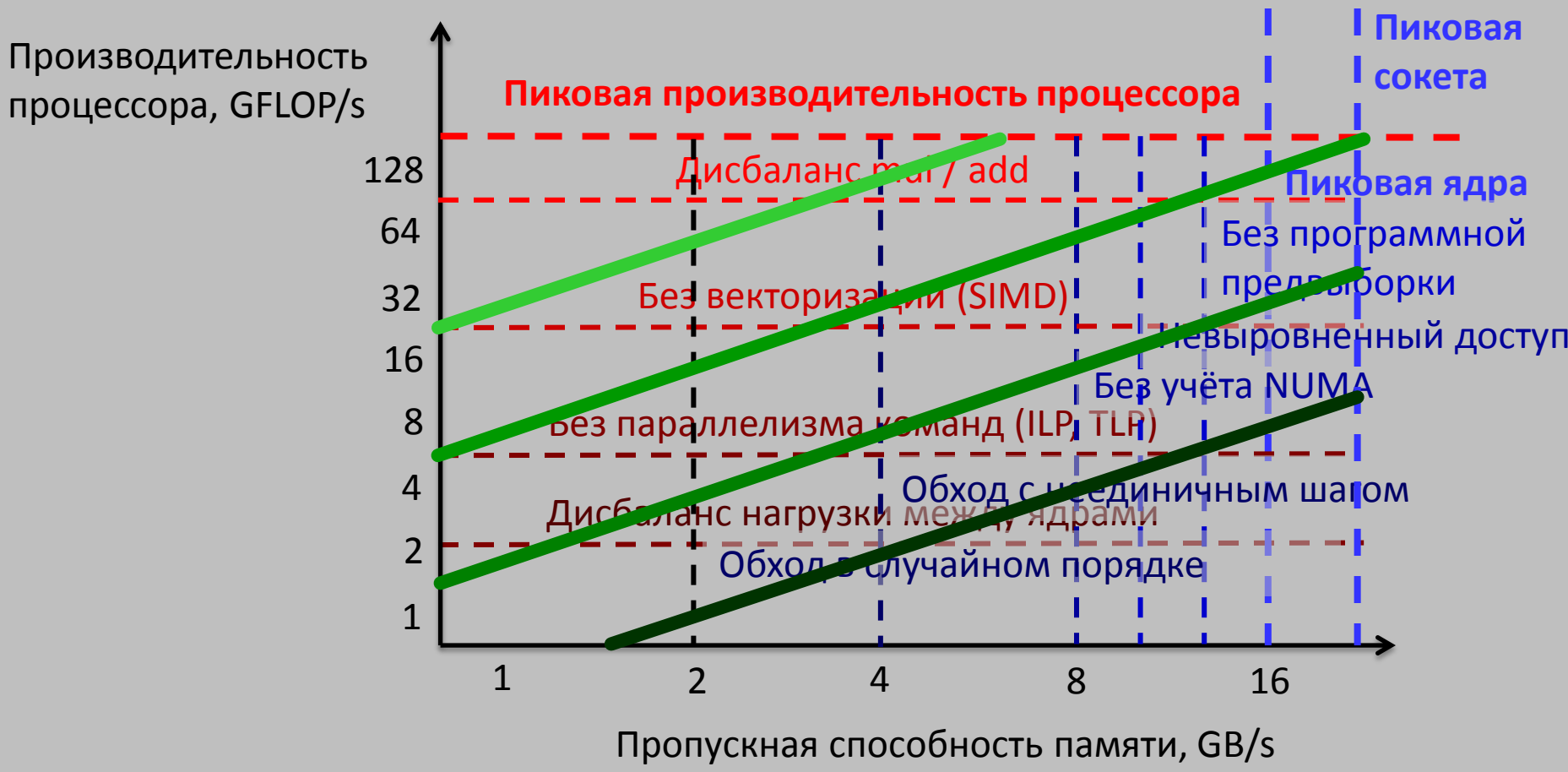
```
for (int i=1;i<N;i++) {
  p *= x;
  k *= i;
  s += p / k;
}
```

- AI = 4*N FLOPS / 0 B = ∞

Границы производительности

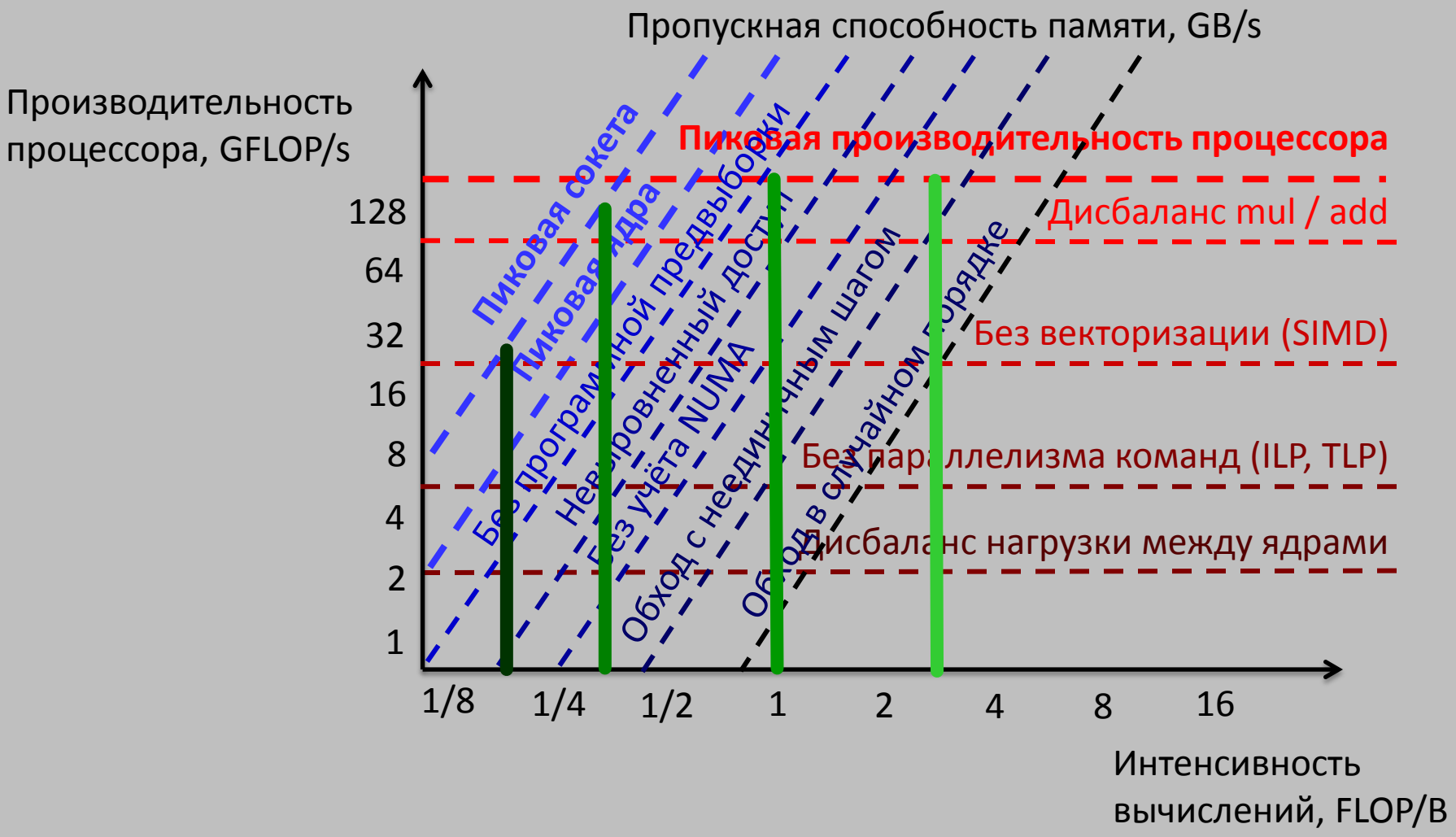


Границы производительности



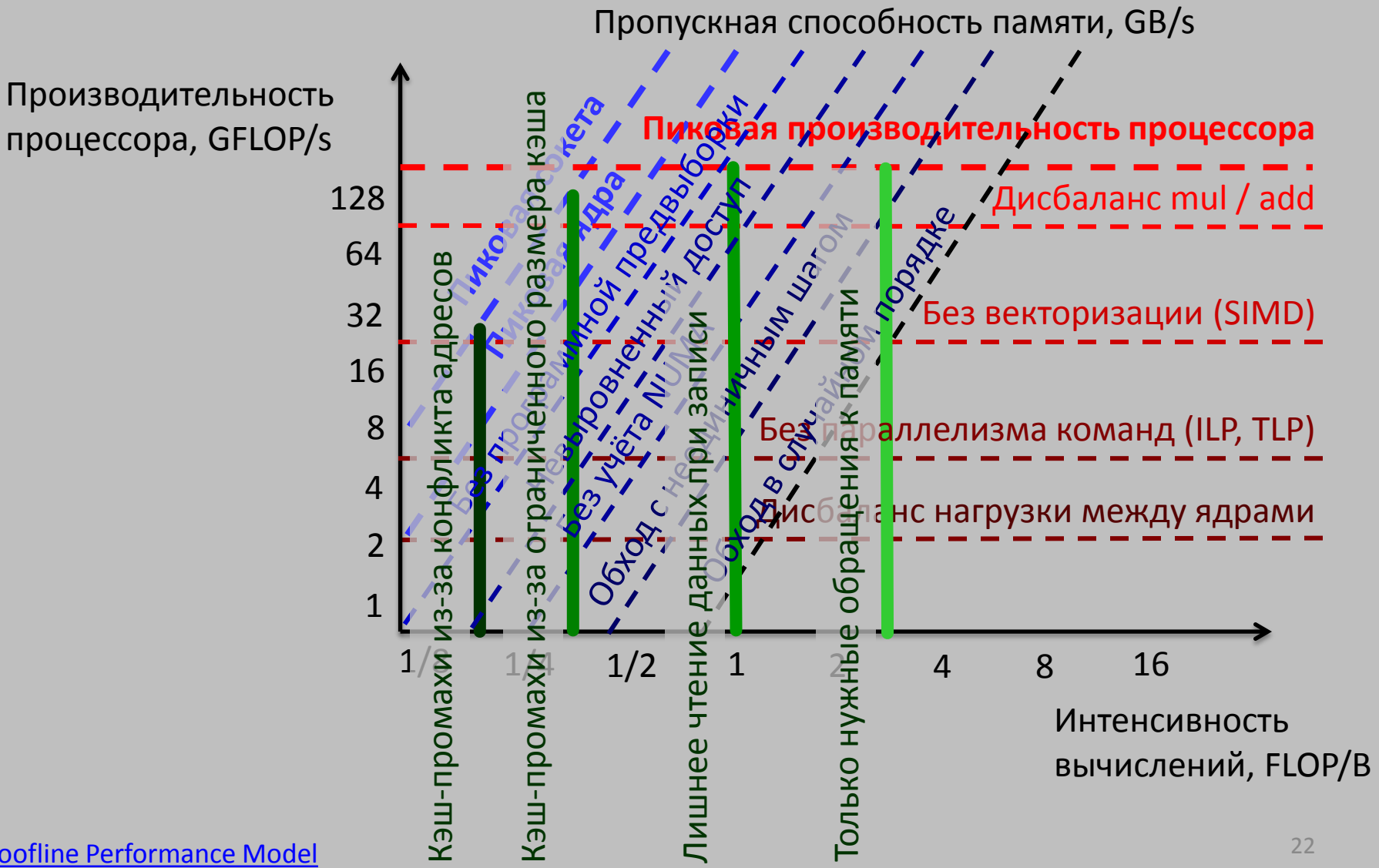
Характеристика программы: Интенсивность вычислений, FLOP/B

Roofline Performance Model



[Roofline Performance Model]

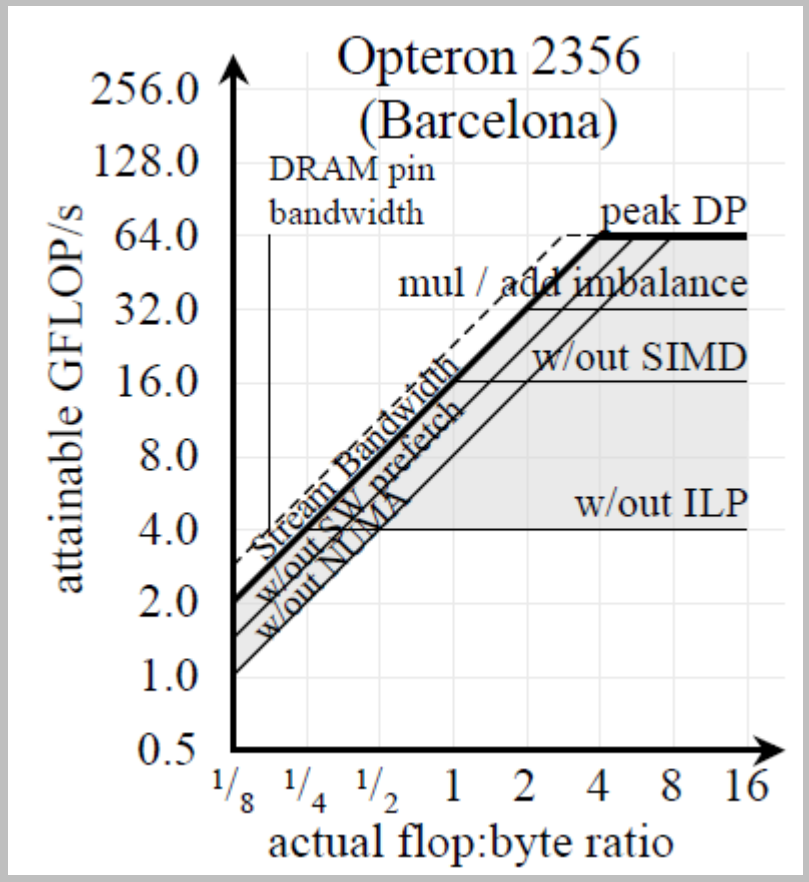
Roofline Performance Model

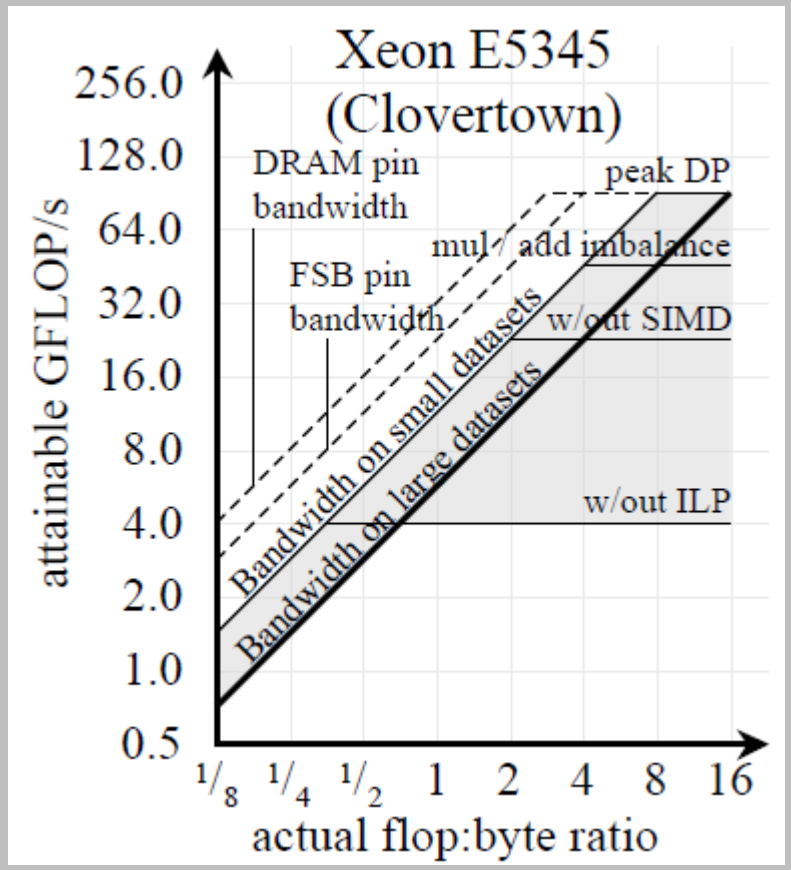


[Roofline Performance Model]

Roofline Performance Model

Примеры анализа архитектур





Три вида программной оптимизации

Увеличение производительности вычислений (FLOPS)

- Использование параллелизма ядра (SIMD, ILP)
- Обеспечение баланса нагрузки между ядрами

Увеличение пропускной способности памяти (MB/s)

- Учёт NUMA
- Сокрытие задержек памяти за вычислениями
- Использование множества запросов в память

Уменьшение количества обращений к памяти (FLOPS/B)

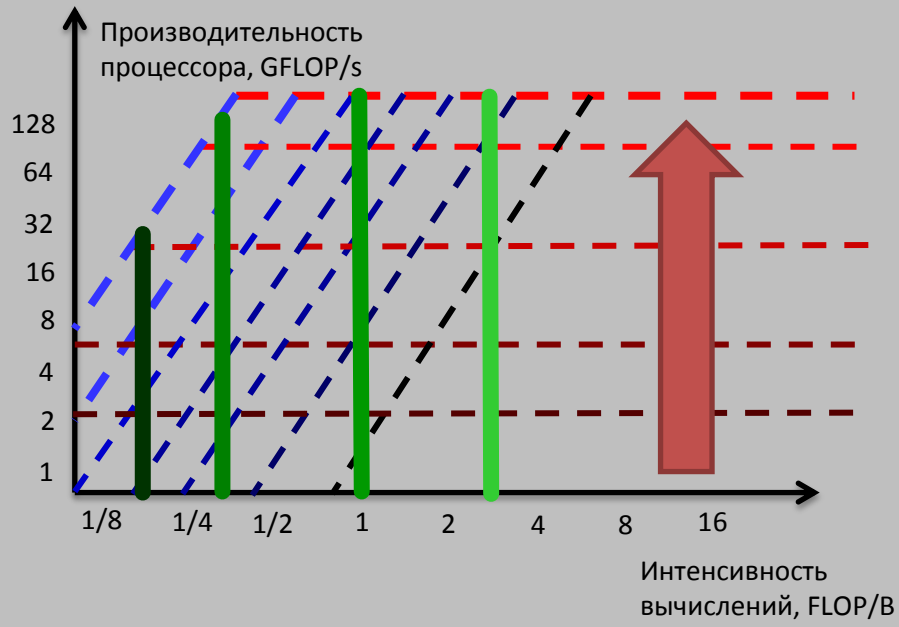
Устранение:

- кэш-промахов
- кэш-буксования
- лишнего чтения памяти при записи

Виды программной оптимизации

Увеличение производительности вычислений

- Использование параллелизма ядра (SIMD, ILP)
- Обеспечение баланса нагрузки между ядрами



Примеры способов оптимизации:

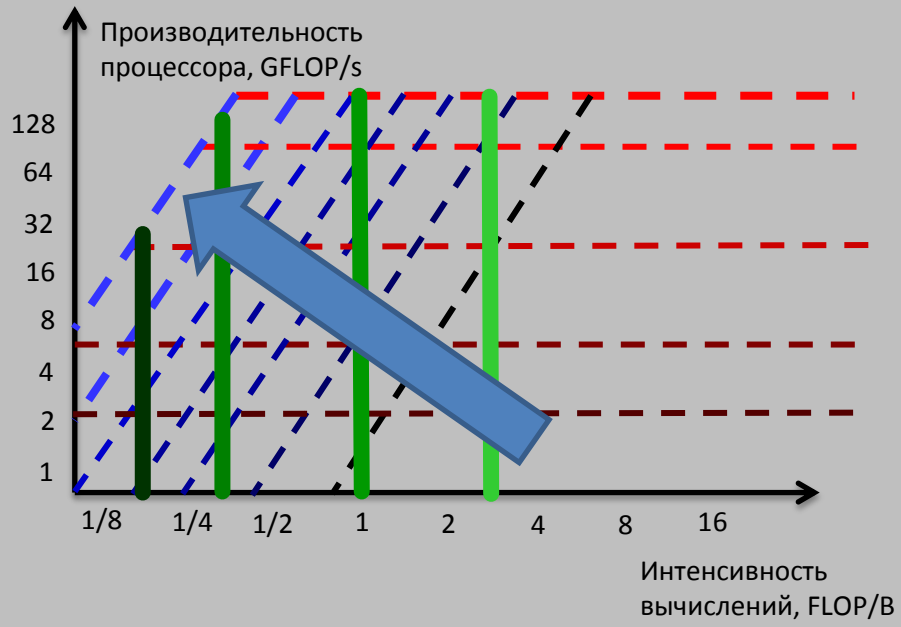
- Раскрутка и слияние циклов
- Программная конвейеризация циклов
- Переупорядочение команд
- Устранение ветвлений
- Векторизация
- …

[Roofline Performance Model]

Виды программной оптимизации

Увеличение пропускной способности памяти

- Учёт NUMA
- Сокрытие задержек памяти за вычислениями
- Использование множества запросов в память



Примеры способов оптимизации:

- Программная предвыборка
- Последовательный обход памяти
- Привязка к ядру
- Использование списков DMA-передач
- …

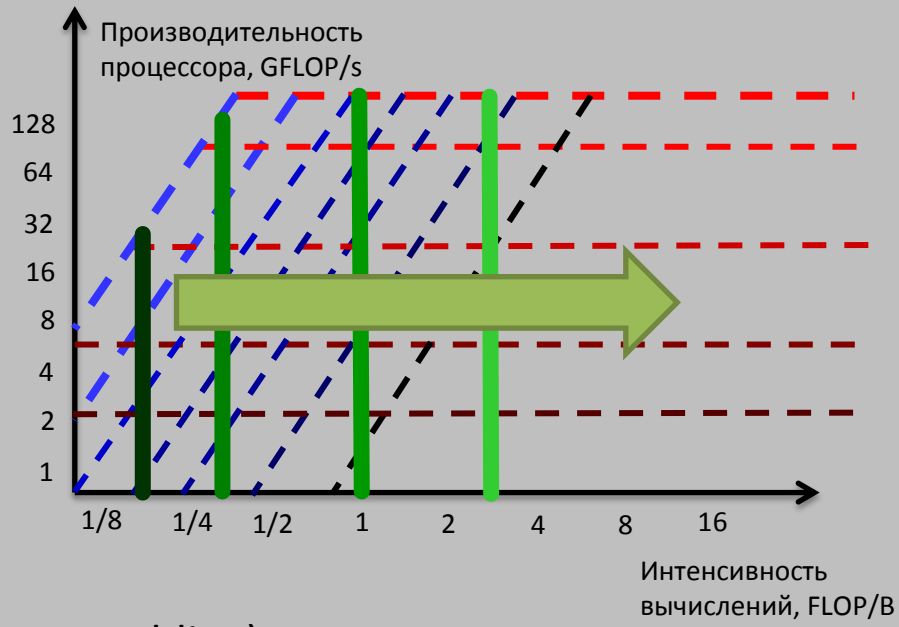
[Roofline Performance Model]

Виды программной оптимизации

Уменьшение количества обращений к памяти

Устранение:

- кэш-промахов
- кэш-буксования
- лишнего чтения памяти при записи



Примеры способов оптимизации:

- Использование блочных алгоритмов
- Относительное смещение массивов (array padding)
- Использование потоковой записи
- Сжатие данных
- …

[Roofline Performance Model]