Международная научная конференция
Параллельные вычислительные технологии (ПаВТ'2018)
Ростов-на-Дону, 3–5 апреля 2018

A Study of Euclidean Distance Matrix Computation on Intel Many-core Processors

Т.В. Речкалов, М.Л. Цымблер

Южно-Уральский государственный университет (Челябинск)

trechkalov@yandex.ru, mzym@susu.ru

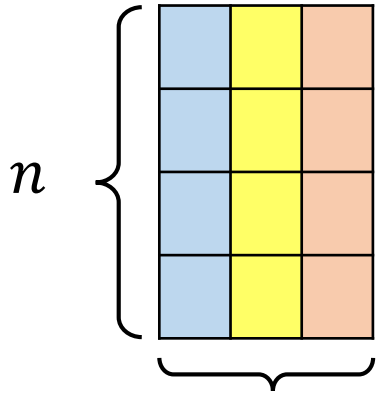
Работа выполнена при финансовой поддержке РФФИ (грант № 17-07-00463), Постановлением Правительства РФ № 211 (договор № 02.A03.21.0011) и Минобрнауки РФ (госзадание 1.9624.2017/7.8).

Применение матрицы Евкл. расстояний

- Анализ ДНК
- Сенсорные сети
- Обработка сигналов
- Извлечение информации из аудио и видео данных
- Алгоритмы кластеризации
- ...

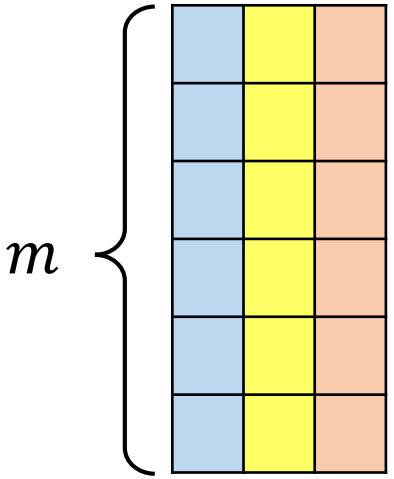
Постановка задачи

$\boldsymbol{A} \in \mathbb{R}^{n \times d}$



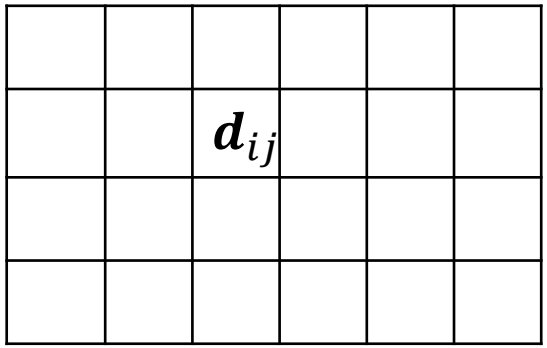
$\boldsymbol{a}_i \in \mathbb{R}^d$

$\boldsymbol{B} \in \mathbb{R}^{m \times d}$

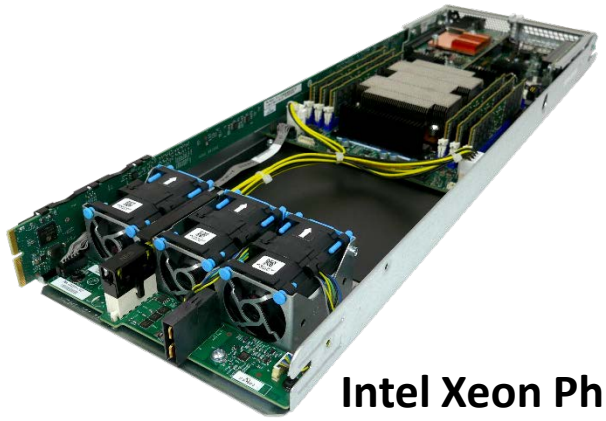


$\boldsymbol{b}_j \in \mathbb{R}^d$

$\boldsymbol{D} \in \mathbb{R}^{n \times m}$



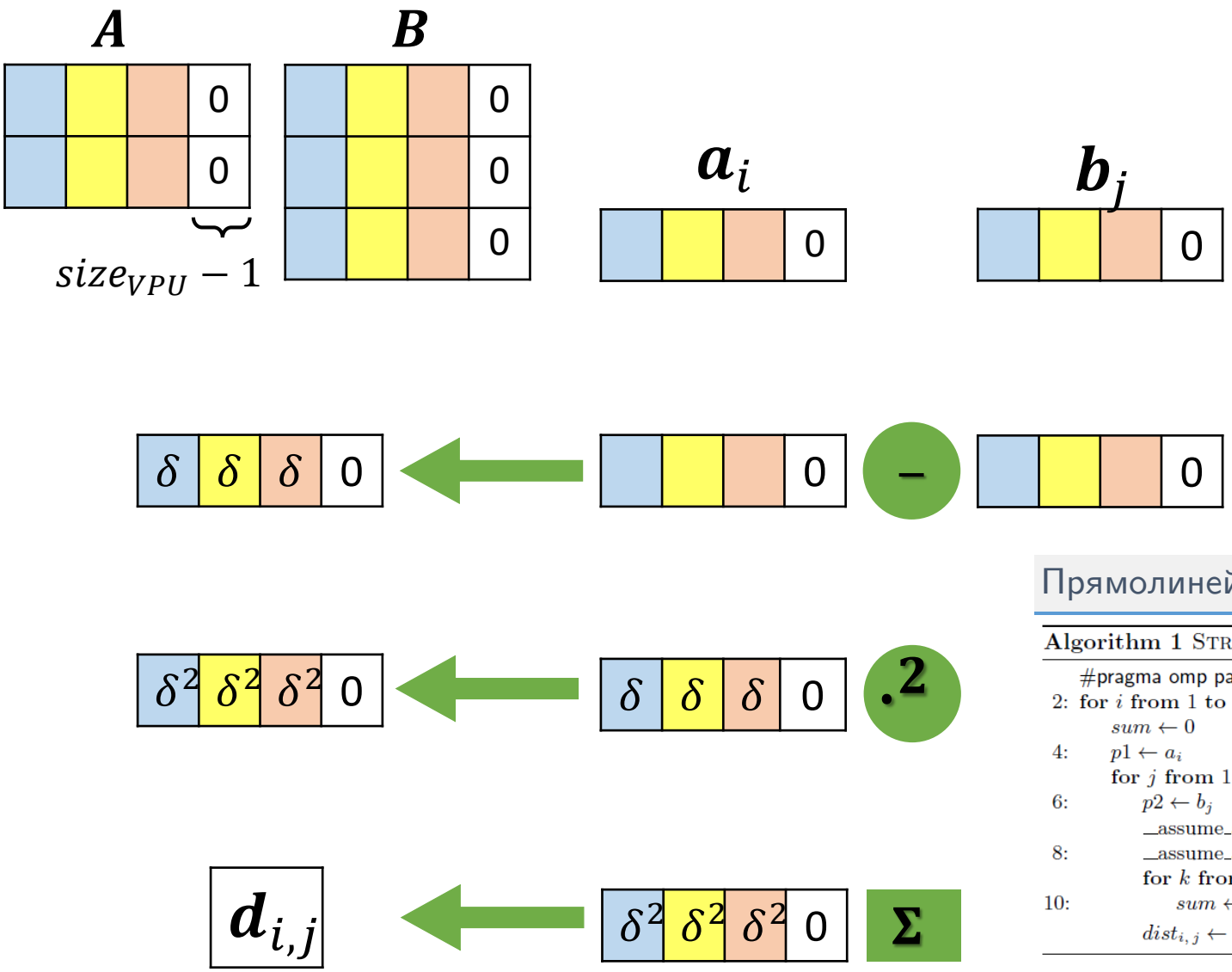
$$\boldsymbol{d}_{ij} = \sum_{k=1}^{d} (\boldsymbol{a}_{i_k} - \boldsymbol{b}_{j_k})^2$$



Intel Xeon Phi

Данные помещаются в память Intel Xeon Phi

Прямолинейный алгоритм для Phi [Lee et. al 2016]



Прямолинейный алгоритм [Lee et. al 2016]

Algorithm 1 StraightforwardEDM(in **A**, **B**; out **D**)

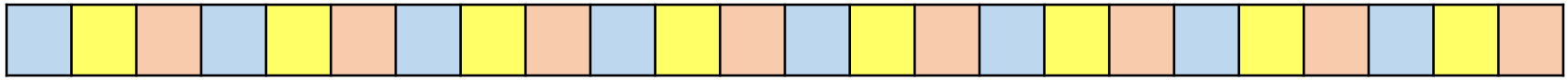
```
   #pragma omp parallel for
2: for i from 1 to n do
       sum ← 0
4:     p1 ← a_i
       for j from 1 to m do
6:         p2 ← b_j
           _assume_aligned(p1, 64)
8:         _assume_aligned(p2, 64)
           for k from 1 to d do
10:            sum ← sum + (p1_k − p2_k)^2
           dist_{i,j} ← sum
```

Размещение данных в памяти

Array of Structures

struct { float x, y, z; } AoS;

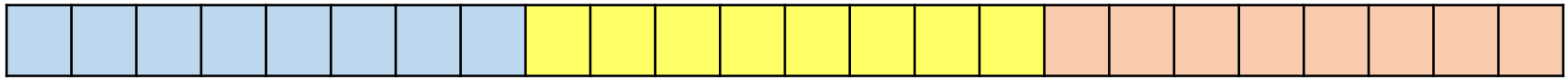
AoS $\boldsymbol{B}[m]$;



Structure of Arrays

struct { float x[m], y[m], z[m]; } SoA;

SoA $\boldsymbol{B}$;



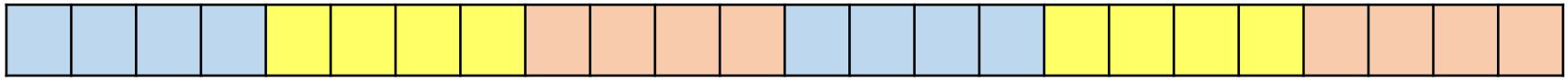
Structure of Arrays of Structures

struct { float x[$size_{block}$], y[$size_{block}$], z[$size_{block}$]; } ASA;

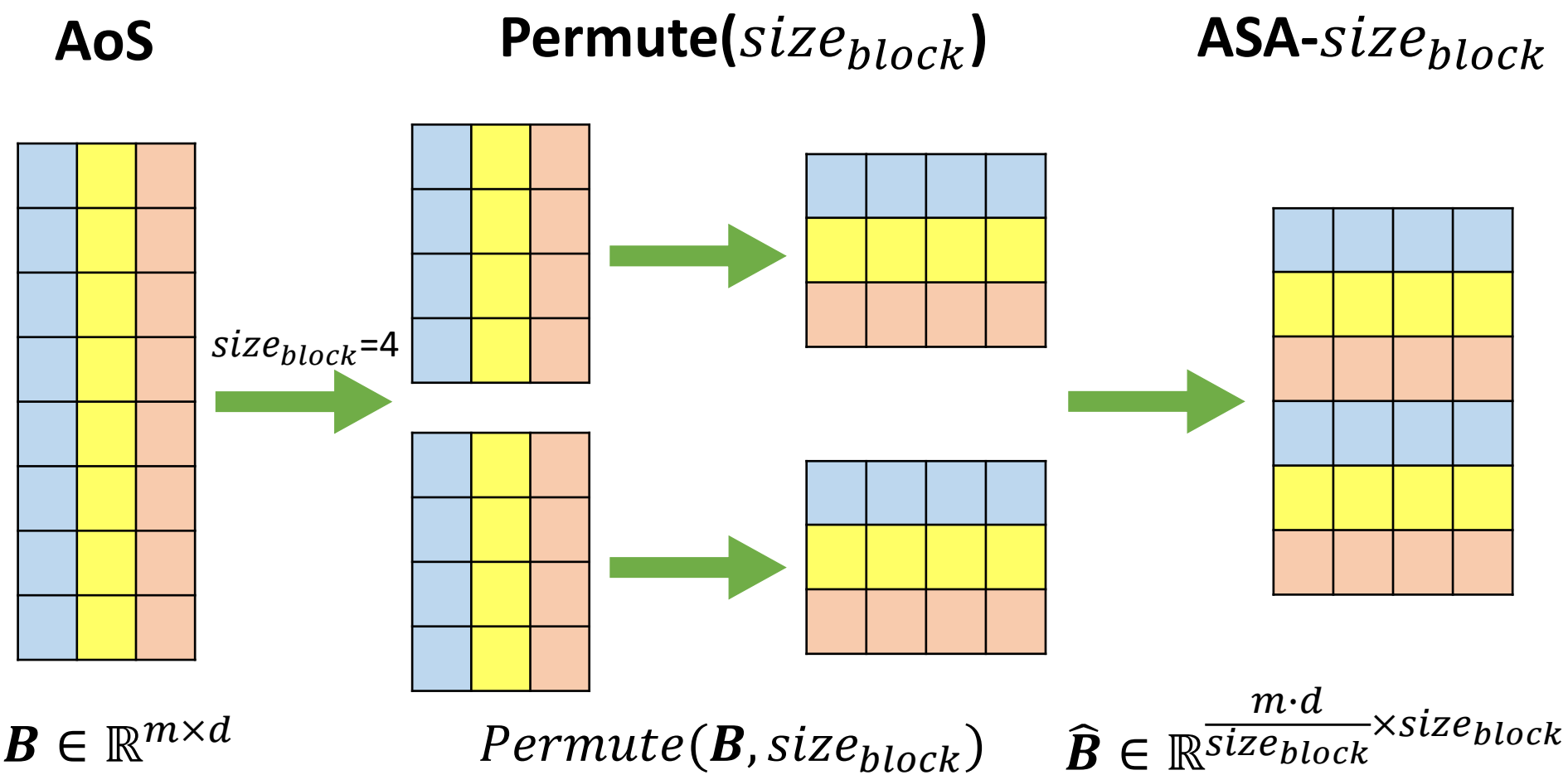
ASA $\boldsymbol{B}[\frac{m}{size_{block}}]$;

$size_{block}$=2

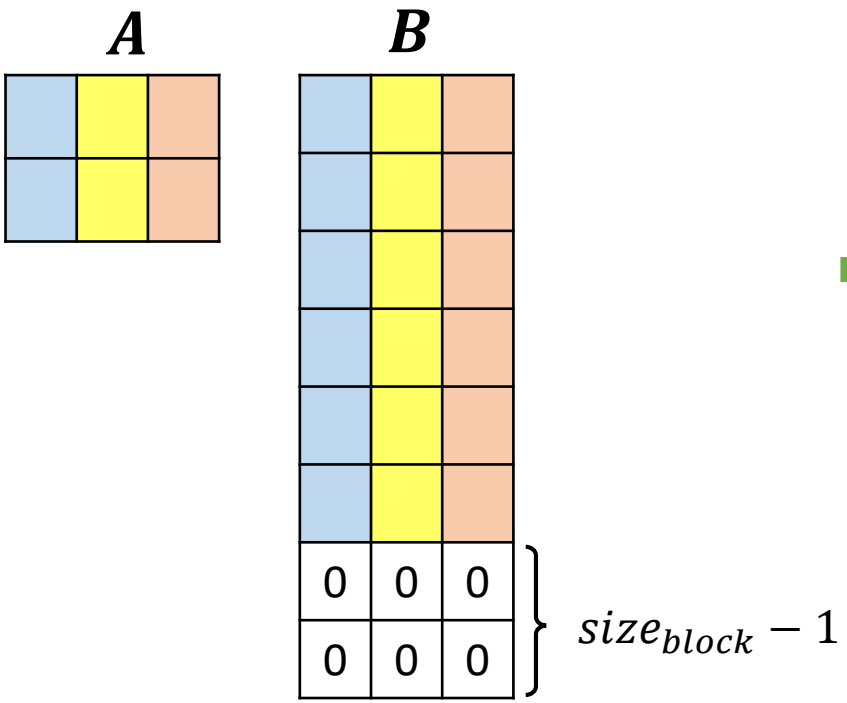
$size_{block}$=4



Перестановка элементов матрицы



Блочный алгоритм для Phi (перестановка)

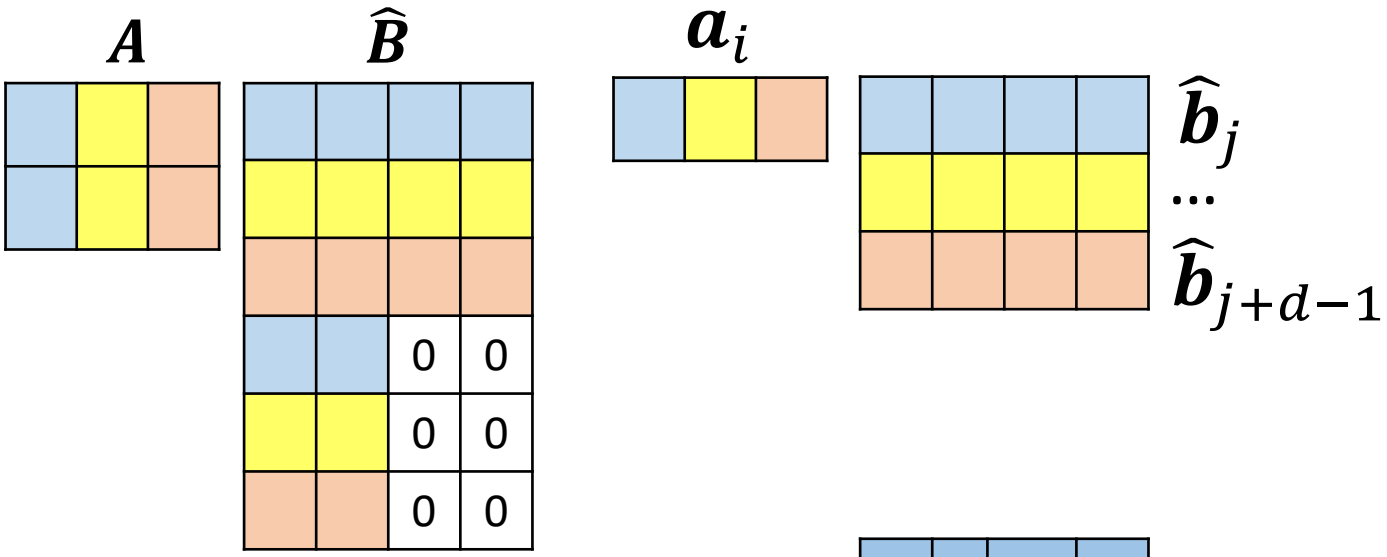


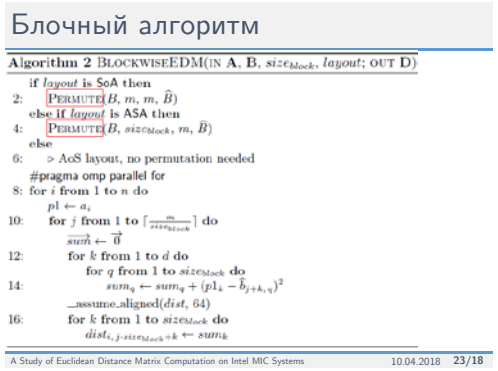
Перестановка элементов матрицы

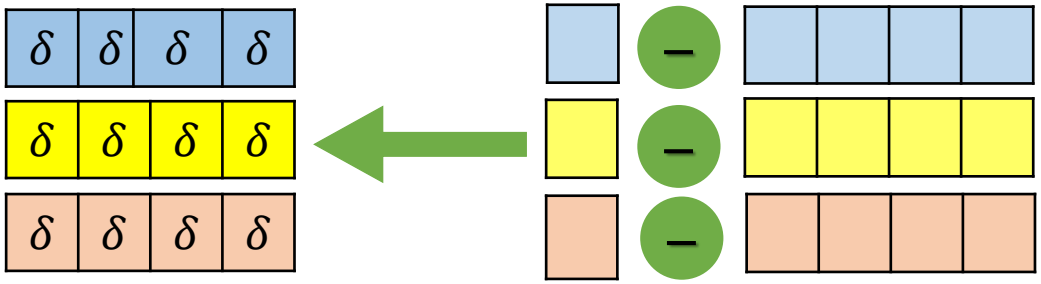
Algorithm 3 PERMUTE(IN **SrcMatr**, $size_{block}$, $rows$; OUT **DstMatr**)

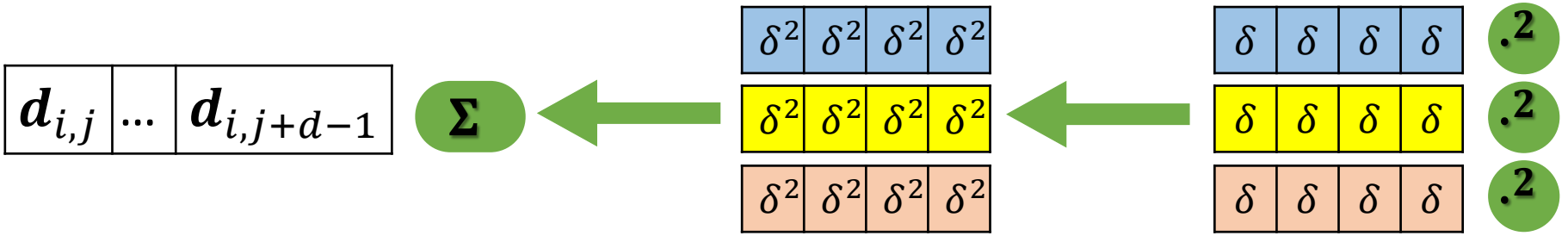
```
    #pragma omp parallel for
2:  for j from 1 to ⌈rows / size_block⌉ do
        for i from 1 to d do
4:          for k from 1 to size_block do
                DstMatr_{j·d+i, k} ← SrcMatr_{j·size_block+k, i}
```

Блочный алгоритм для Phi (вычисление)



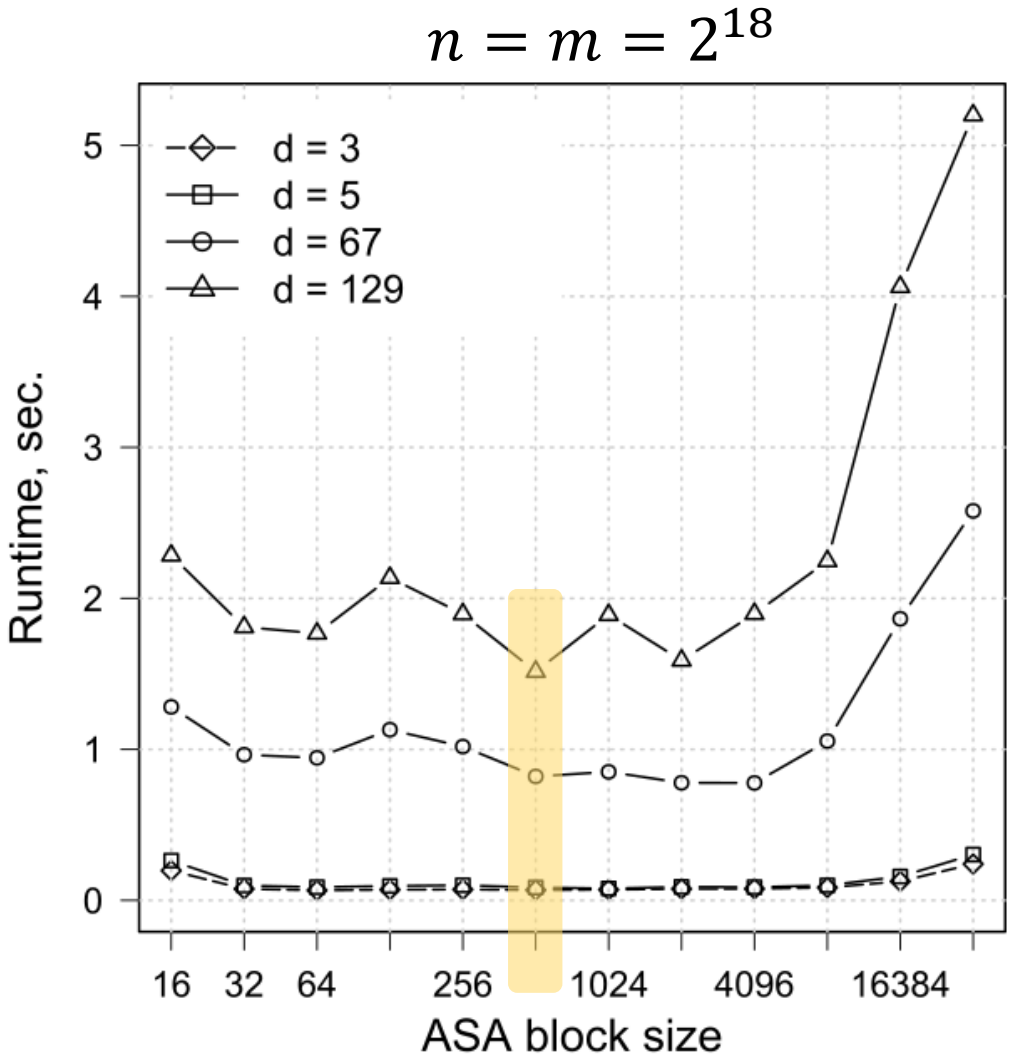






Выбор размера блока

- $size_{block} : size_{VPU}$
- $m : size_{block}$
- Эмпирически:
  $size_{block}$=512



Эксперименты

- Аппаратная платформа

Платформа, Intel / Характеристика	2×Xeon X5680	Xeon Phi SE10X
Кол-во физ. ядер	2×6	61
Гиперпоточность	2	4
Кол-во лог. ядер	2×12	244
Тактовая частота, GHz	3.33	1.1
Оперативная память, Gb	16	8
Пик. произв-ть, TFLOPS	0.371	1.076

- Цели
  - Быстродействие, ускорение, параллельная эффективность
  - Сравнение с прямолинейным алгоритмом
  - Сравнение на Intel Xeon Phi и 2$\times$Intel Xeon
  - Сравнение с алгоритмом из Intel MKL

Квадратные матрицы

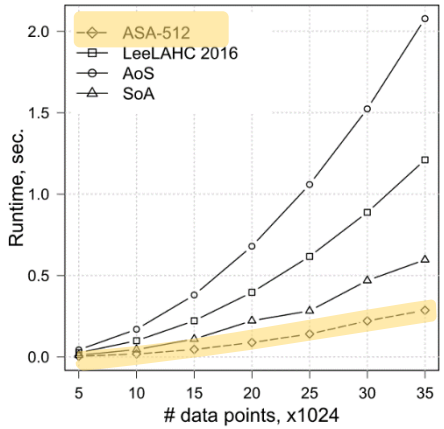
Набор	d	$n = m$, $\times 2^{10}$	Семантика
MixSim	5	35	Синтетический генератор тестов для оценки алгоритмов кластеризации [Melnykov et al. 2012]
Census	67	35	Отчеты Бюро переписи населения США [Meek et. al 2002]
FCS Human	423	18	Генетические данные [Engreitz et. al 2010]

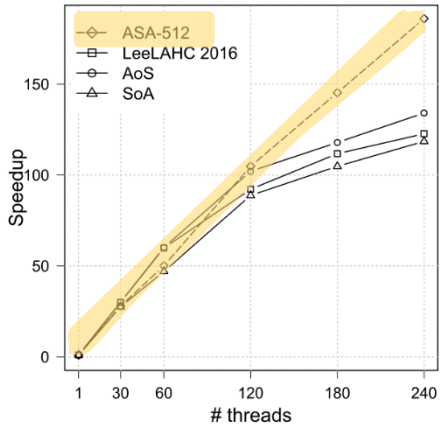
- Для запуска прямолинейного и блочного алгоритмов каждая точка данных дополняется нулями до ближайшего к d целого, кратного размеру векторного регистра (т.е. d'=16, 80, 432)

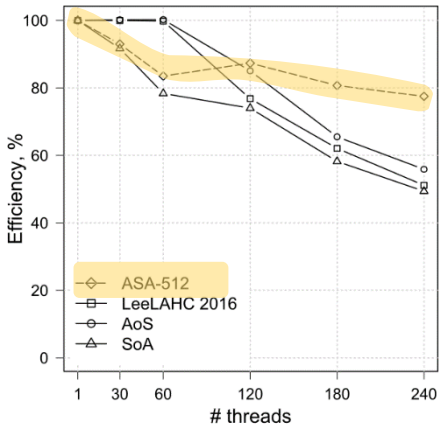
Квадратные матрицы

MixSim

d	$n = m$, $\times 2^{10}$
16	35

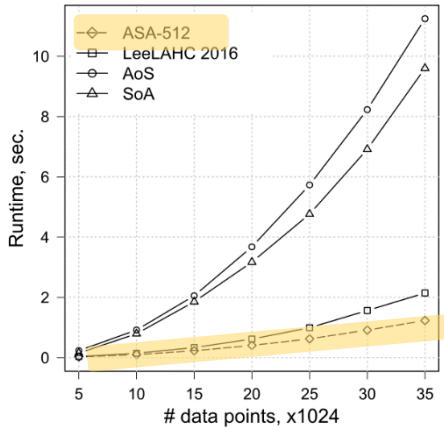


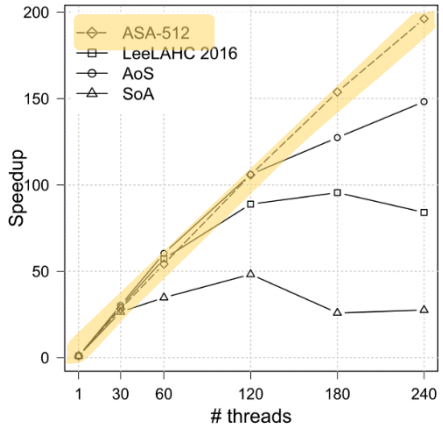


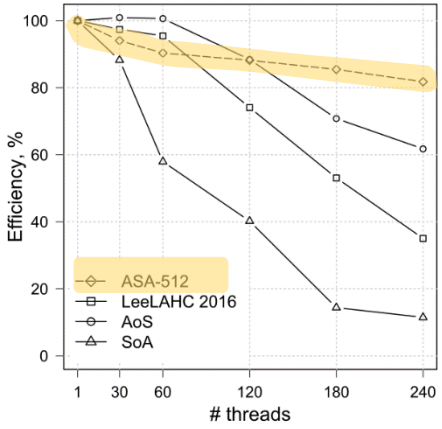


Census

d	$n = m$, $\times 2^{10}$
80	35

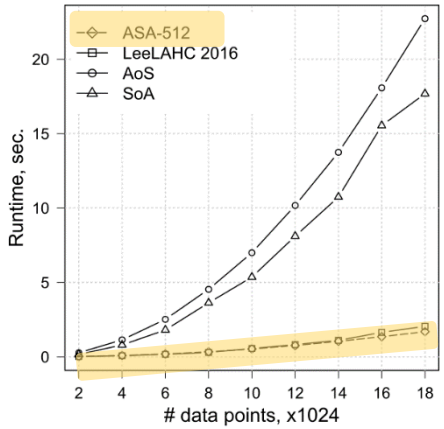


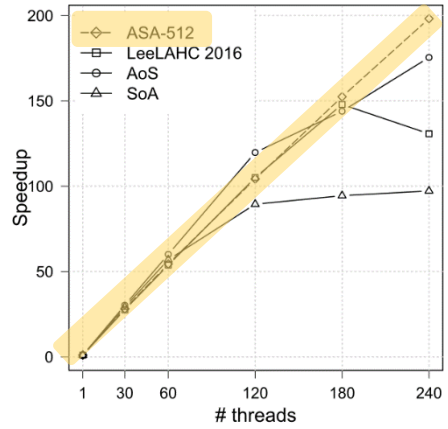


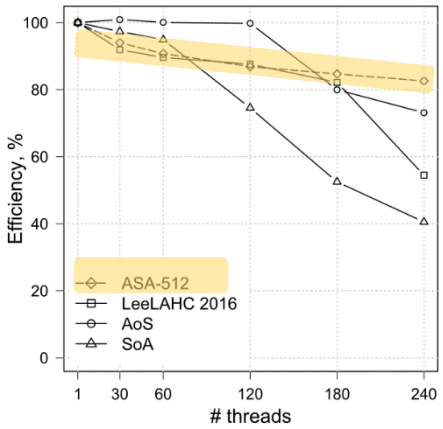


FCS Human

d	$n = m$, $\times 2^{10}$
432	18







Прямоугольные матрицы

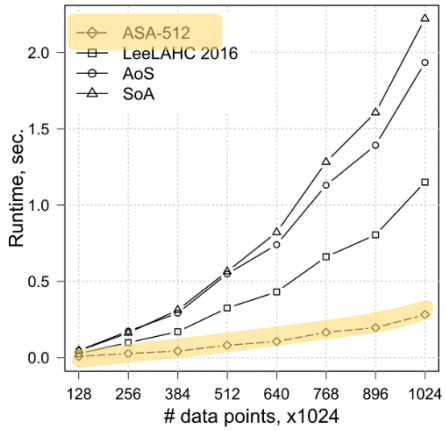
Набор	d	n	m	Семантика
ADS-16	16	10^3	10^6	Синтетические данные, использованные в [Lee et al. 2016]
ADS-32	32			
ADS-64	64			
ADS-128	128			
ADS-256	256			

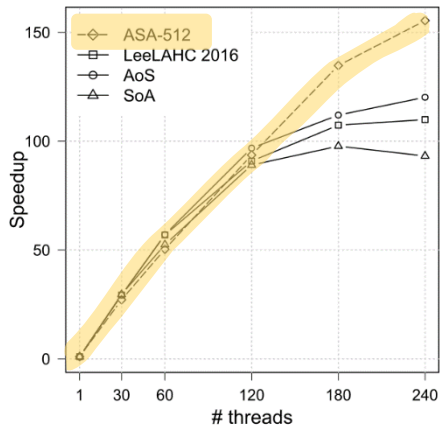
- Для запуска блочного алгоритма каждое множество точек дополняется нулевыми точками до ближайшего к m целого, кратного размеру блока (т.е. $m'=2^{20}$)

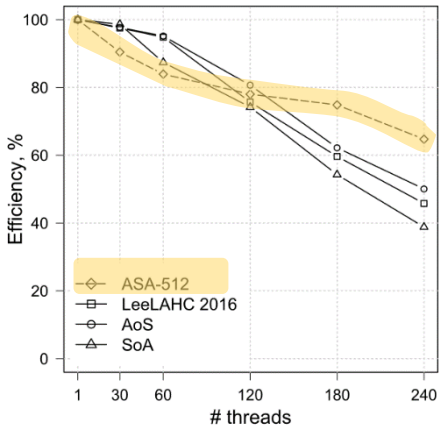
Прямоугольные матрицы

ADS-16

d	n	m
16	10^3	2^{20}

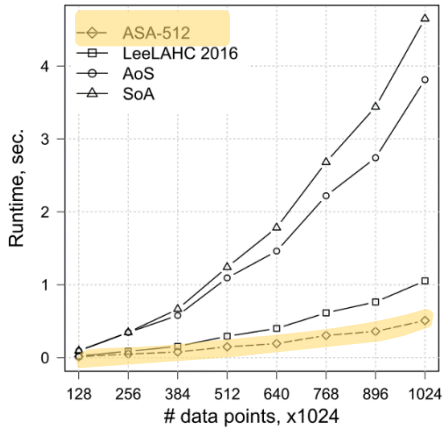


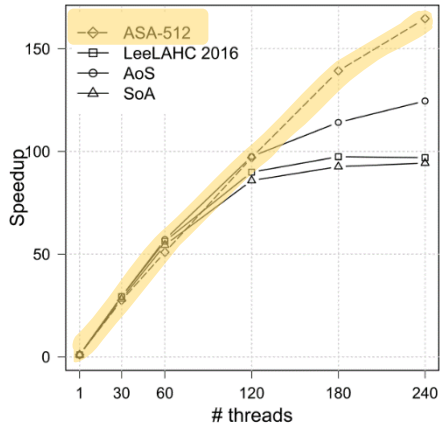


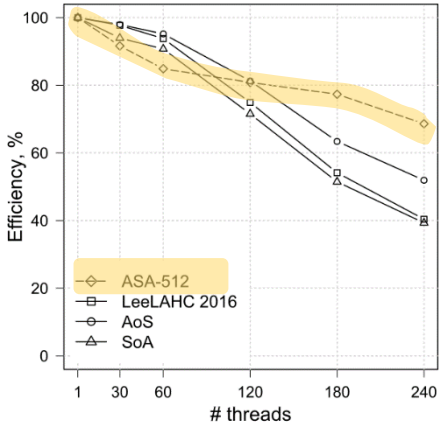


ADS-32

d	n	m
32	10^3	2^{20}

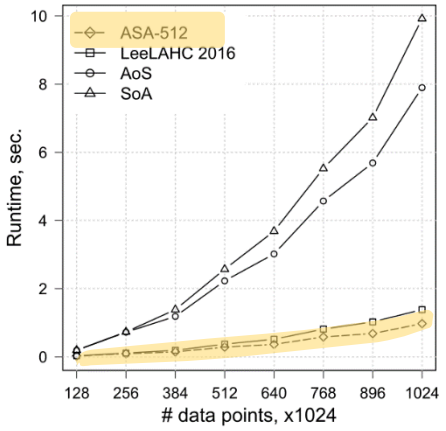


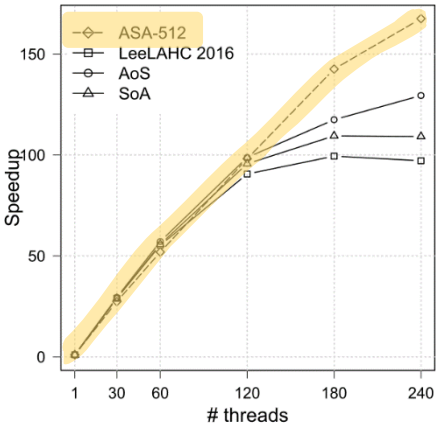


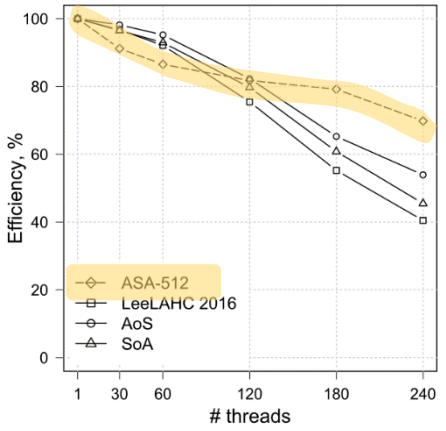


ADS-64

d	n	m
64	10^3	2^{20}



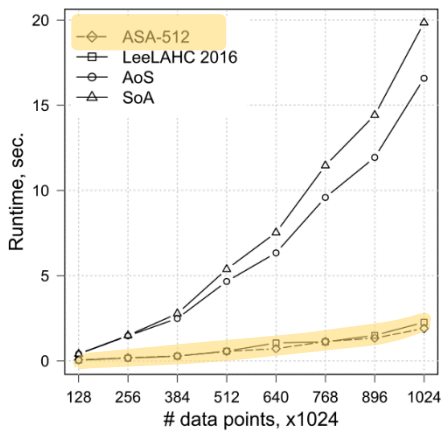


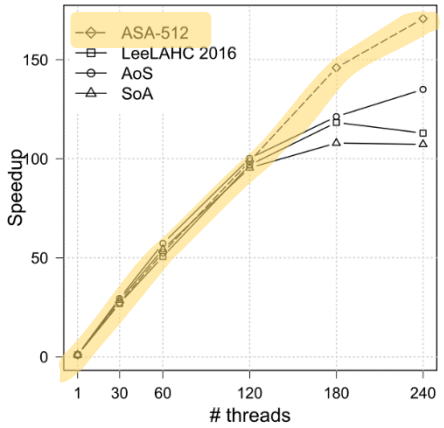


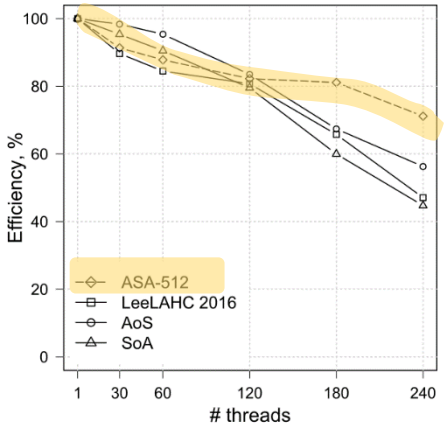
Прямоугольные матрицы

ADS-128

d	n	m
128	10^3	2^{20}

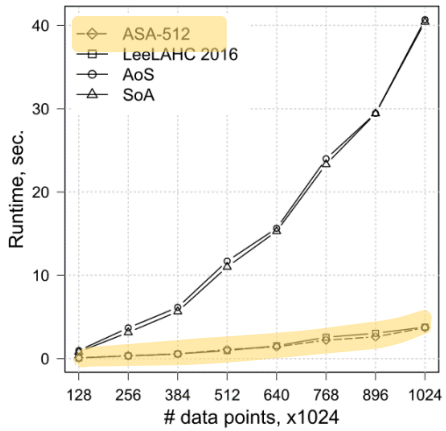


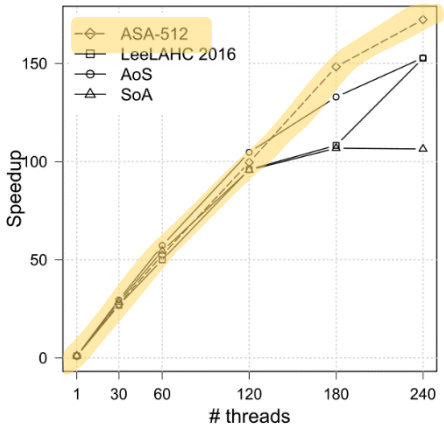


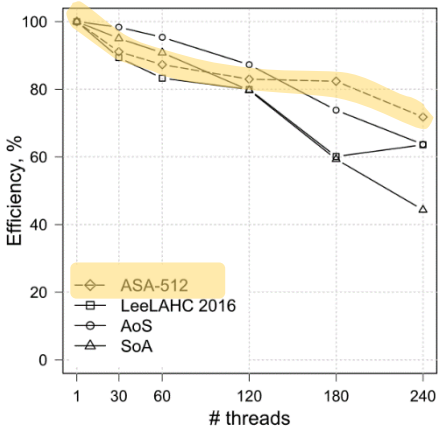


ADS-256

d	n	m
256	10^3	2^{20}





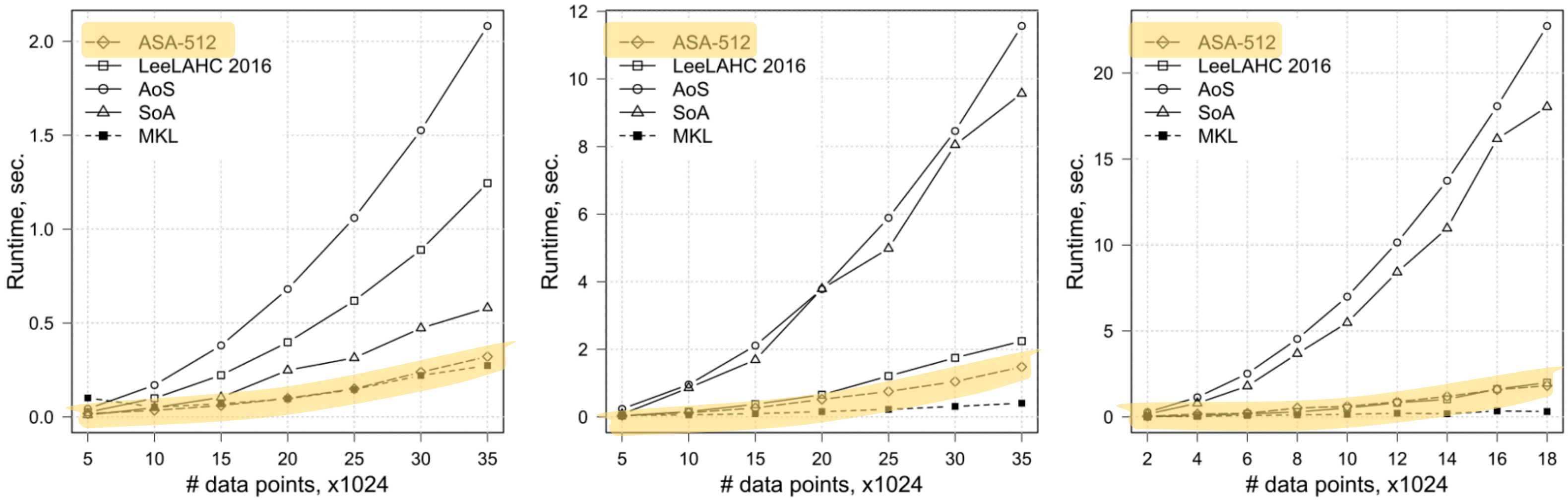


Сравнение с 2×Intel Xeon

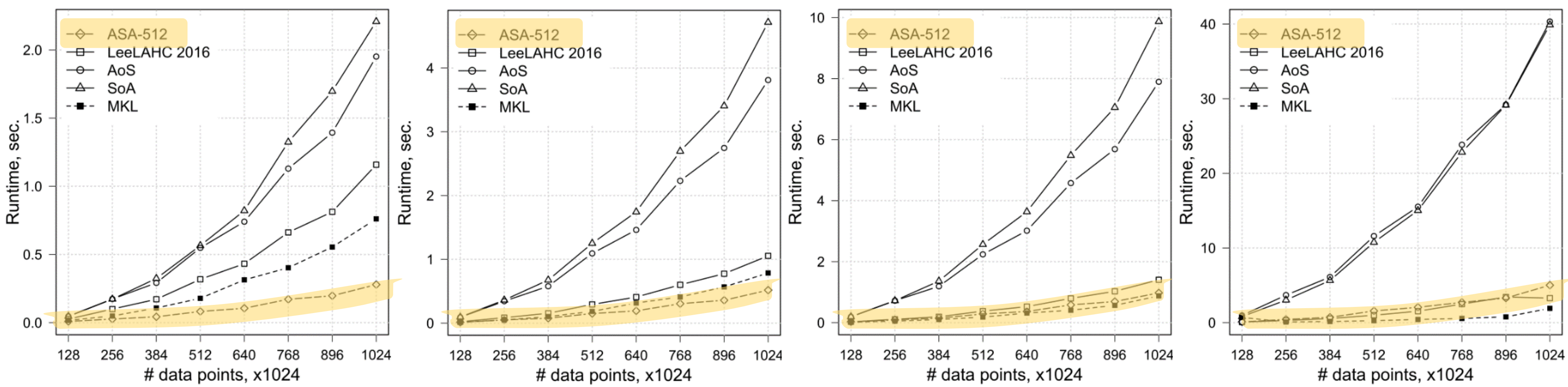
Набор	Intel Xeon Phi, s		2×Intel Xeon, s		CPU/Phi	
	ASA-512	Lee et al.	ASA-512	Lee et al.	ASA-512	Lee et al.
ADS-16	0.28	1.15	1.04	1.00	3.7×	1.0×
ADS-32	0.51	1.05	1.76	1.79	3.5×	1.7×
ADS-64	0.98	1.36	3.78	4.25	3.9×	3.1×
ADS-256	3.71	3.79	30.32	31.41	8.2×	8.3×

Сравнение с Intel MKL

Набор	d	$n = m, \times 2^{10}$
MixSim	16	35
Census	80	35
FCS	432	18



Набор	d	n	m
ADS-16	16	2^{10}	2^{20}
ADS-32	32		
ADS-64	64		
ADS-256	256		



Заключение

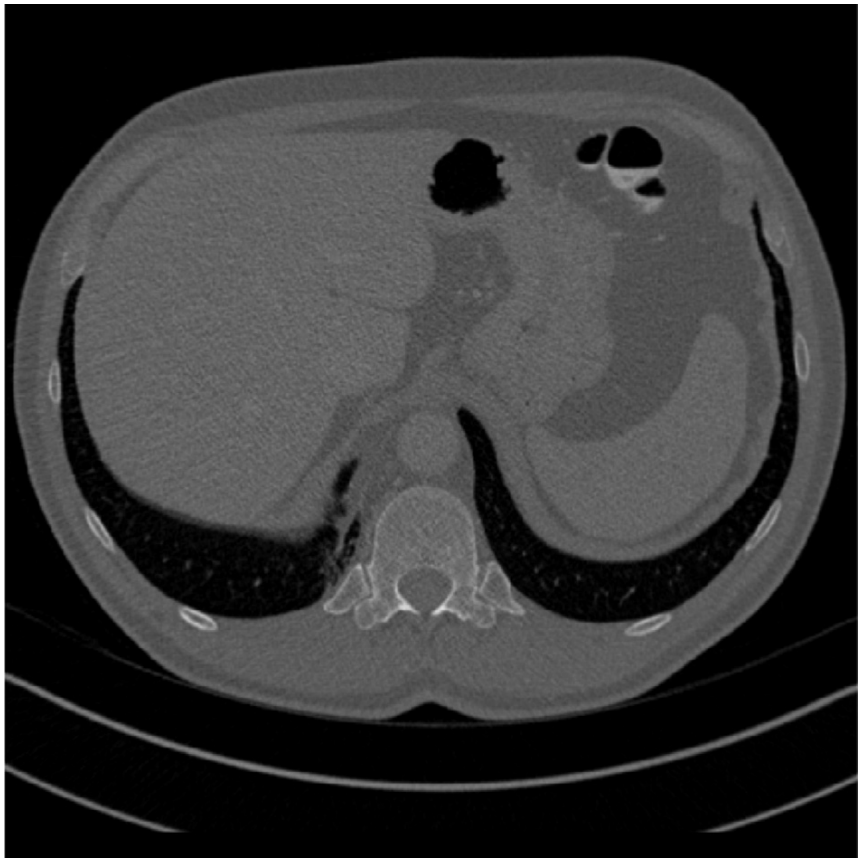
- Предложена новая схема нахождения матрицы Евклидовых расстояний для улучшения векторизации вычислений на процессоре Intel Xeon Phi
- Проведены эксперименты, показавшие эффективность разработки для данных малой размерности
- Будущие исследования
  - Применение решения в алгоритмах кластеризации
  - Аналитический подбор значения $size_{block}$
  - Распределенная версия

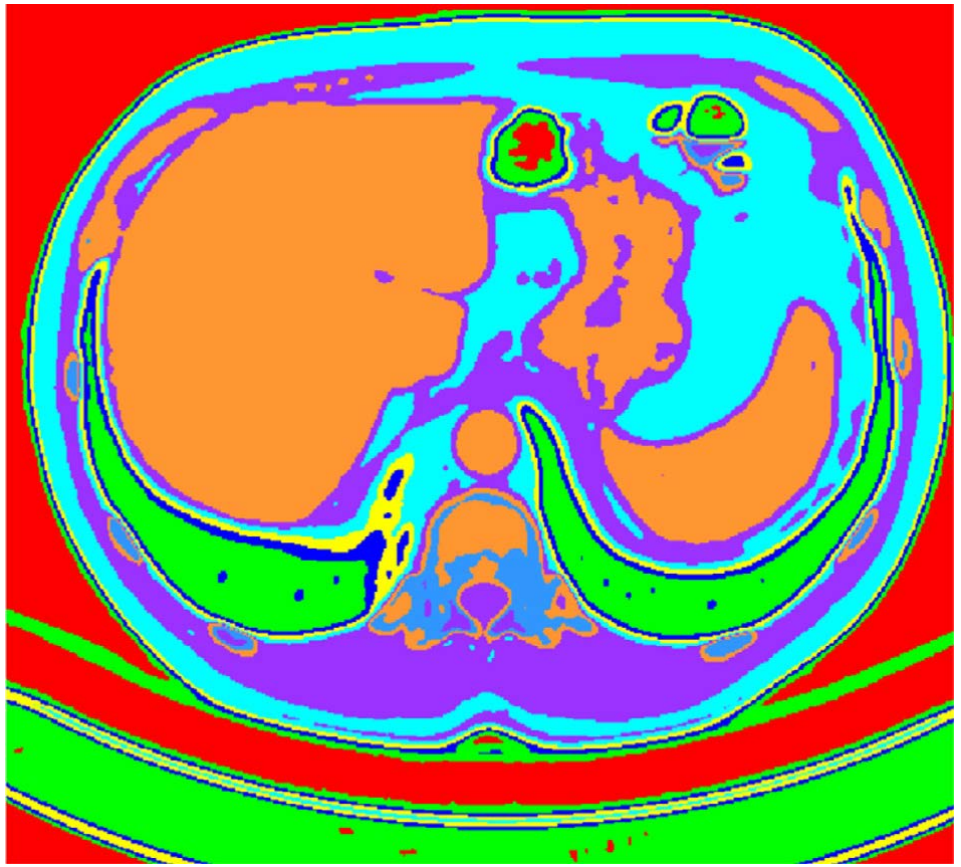
Спасибо за внимание! Вопросы?

Михаил Леонидович Цымблер

mzym@susu.ru

Сегментирование мед. изображений

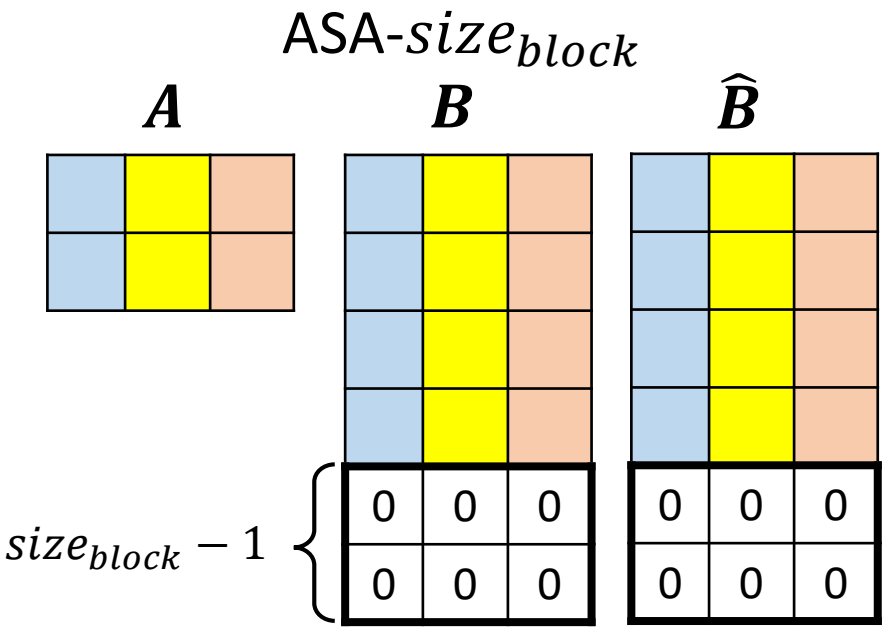


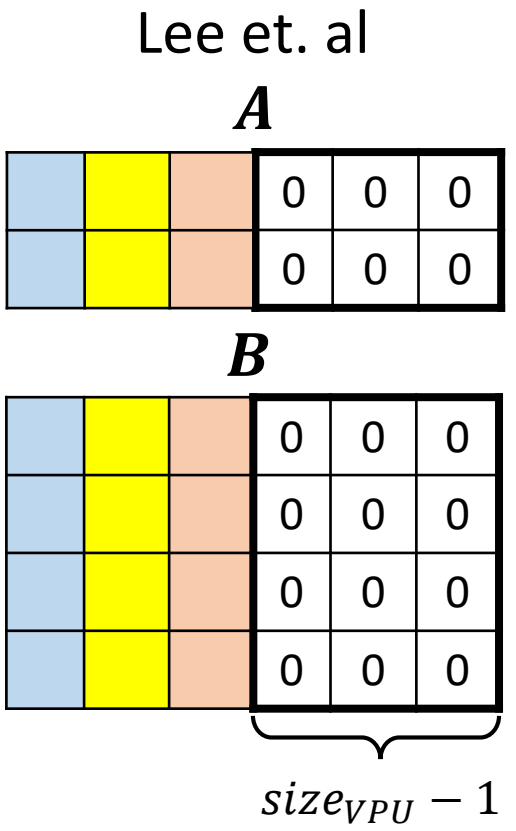


- Кластеризация точек снимков компьютерного томографа (снимок – 512×512 4-мерных точек, 2000 снимков)

Накладные расходы

- Память





- Вычисления
  - Перестановка: ≈1% от времени счета
  - Эмпирический подбор $size_{block}$

Сравнение с GPU

Набор	d	n	m	Семантика
PRND-50	50	$15 \cdot 10^3$	$15 \cdot 10^3$	Синтетические данные, использованные в [Kim et al. 2012]
PRND-100	100			
PRND-150	150			
PRND-200	200			

Набор	**Intel Xeon Phi**, s 1.076 TFLOPS	**2× Intel Xeon**, s 0.371 TFLOPS	**NVIDIA Tesla C2050**, s 1.03 TFLOPS
	ASA-512	**ASA-512**	**Kim et al.**
PRND-50	0.19	0.35	0.82
PRND-100	0.32	0.59	1.01
PRND-150	0.45	0.78	1.21
PRND-200	0.58	1.60	1.41

Прямолинейный алгоритм [Lee et. al 2016]

Algorithm 1 StraightforwardEDM(in **A**, **B**; out **D**)

```
    #pragma omp parallel for
 2: for i from 1 to n do
        sum ← 0
 4:     p1 ← a_i
        for j from 1 to m do
 6:         p2 ← b_j
            __assume_aligned(p1, 64)
 8:         __assume_aligned(p2, 64)
            for k from 1 to d do
10:             sum ← sum + (p1_k − p2_k)^2
            dist_{i,j} ← sum
```

Блочный алгоритм

Algorithm 2 BLOCKWISEEDM(IN **A**, **B**, $size_{block}$, *layout*; OUT **D**)

```
    if layout is SoA then
2:      PERMUTE(B, m, m, B̂)
    else if layout is ASA then
4:      PERMUTE(B, size_block, m, B̂)
    else
6:      ▷ AoS layout, no permutation needed
    #pragma omp parallel for
8:  for i from 1 to n do
        p1 ← a_i
10:     for j from 1 to ⌈m / size_block⌉ do
            sum⃗ ← 0⃗
12:         for k from 1 to d do
                for q from 1 to size_block do
14:                 sum_q ← sum_q + (p1_k − b̂_{j+k,q})^2
            __assume_aligned(dist, 64)
16:         for k from 1 to size_block do
                dist_{i, j·size_block+k} ← sum_k
```

Перестановка элементов матрицы

Algorithm 3 PERMUTE(IN **SrcMatr**, $size_{block}$, $rows$; OUT **DstMatr**)

#pragma omp parallel for
2: **for** j **from** 1 **to** $\lceil \frac{rows}{size_{block}} \rceil$ **do**
  for i **from** 1 **to** d **do**
4: **for** k **from** 1 **to** $size_{block}$ **do**
    $DstMatr_{j \cdot d+i,\, k} \leftarrow SrcMatr_{j \cdot size_{block}+k,\, i}$