ИНТЕЛЛЕКТУАЛЬНЫЙ АНАЛИЗ ДАННЫХ В СУБД

05.13.11 – математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей

Диссертация на соискание ученой степени доктора физико-математических наук

Михаил Леонидович Цымблер

Научный консультант:
Соколинский Леонид Борисович,
доктор физ.-мат. наук, профессор

Цель диссертационной работы

Разработка комплекса масштабируемых методов и параллельных алгоритмов для создания программной платформы интеллектуального анализа данных средствами СУБД с открытым кодом

Актуальность исследования

- Приближение алгоритмов к данным
  - технологии СУБД для больших данных
  - интеграция анализа данных в СУБД
- Импортозамещение
  - российские СУБД и системы анализа данных
  - решения на основе СУБД с открытым кодом

Основные задачи

1. Разработать методы и алгоритмы для внедрения параллелизма в свободную последовательную СУБД
2. Разработать, реализовать и исследовать комплекс параллельных алгоритмов интеллектуального анализа данных на основе параллельной СУБД

Комплекс алгоритмов интеллектуального анализа данных

- Алгоритмы кластеризации данных
  - Кластеризация графов
  - Нечеткая кластеризация
  - Кластеризация данных с шумами и выбросами
- Алгоритмы анализа временных рядов
  - Поиск похожих подпоследовательностей
  - Поиск аномальных подпоследовательностей
- Алгоритмы поиска шаблонов

Организация параллельной обработки

Отношение R фрагментировано по атрибуту ID

$\varphi: R \to \{0, \ldots, 9\}$

$\varphi(r) = r.ID \; div \; 10 \; mod \; 10$

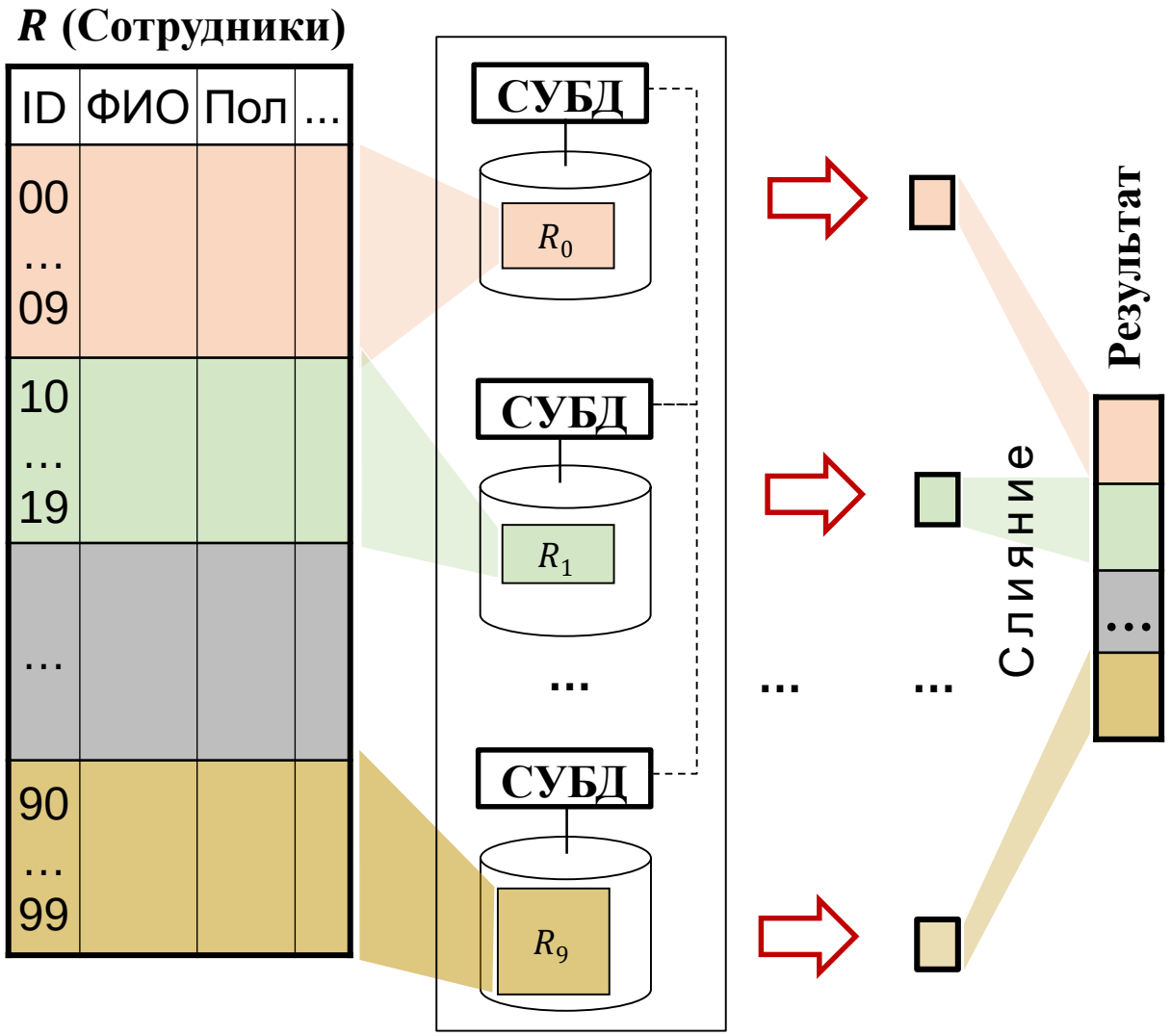
Фрагмент отношения

$R_i = \{r \mid r \in R, \varphi(r) = i\}$,

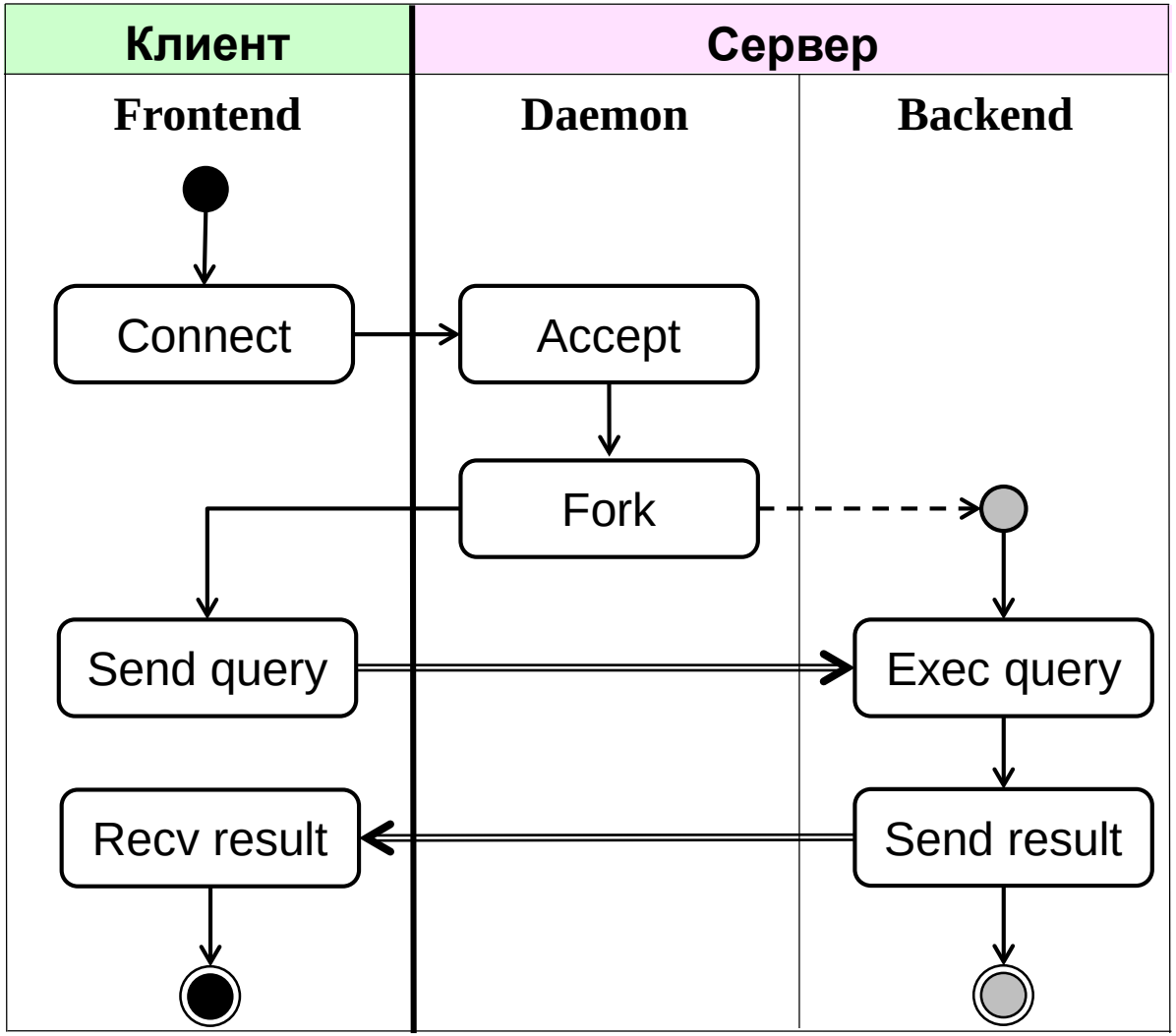
$0 \le i < P$,

где P – количество серверов

```
SELECT ФИО
FROM R
WHERE Пол='M'
```



Процессы PostgreSQL



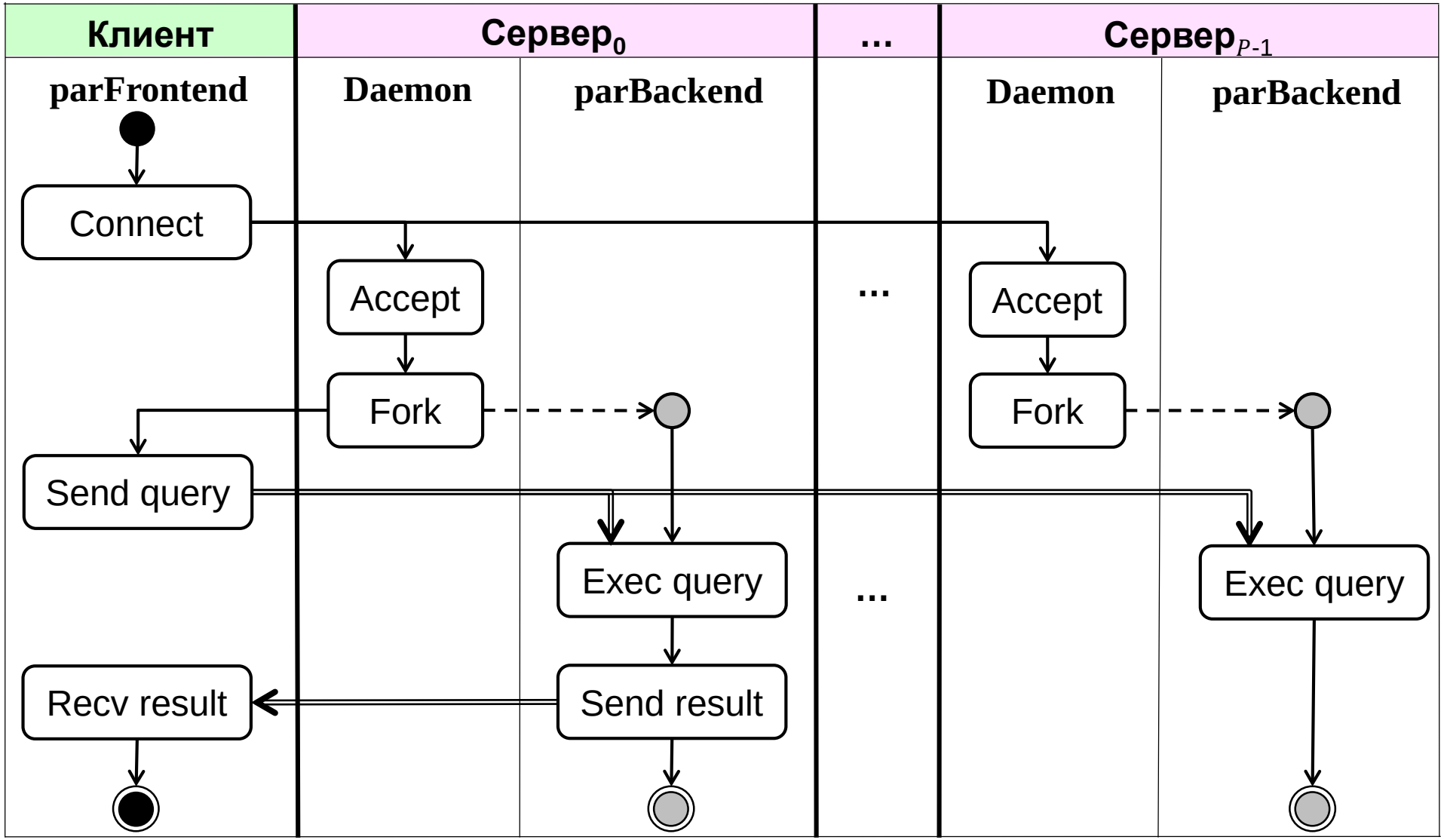
- **Frontend**
  приложение-клиент с подключенной библиотекой PostgreSQL
- **Daemon**
  серверный процесс, который принимает соединение клиента и создает экземпляр Backend для обработки запросов клиента
- **Backend**
  серверный процесс, который выполняет запрос клиента и отправляет результат клиенту

Передача управления

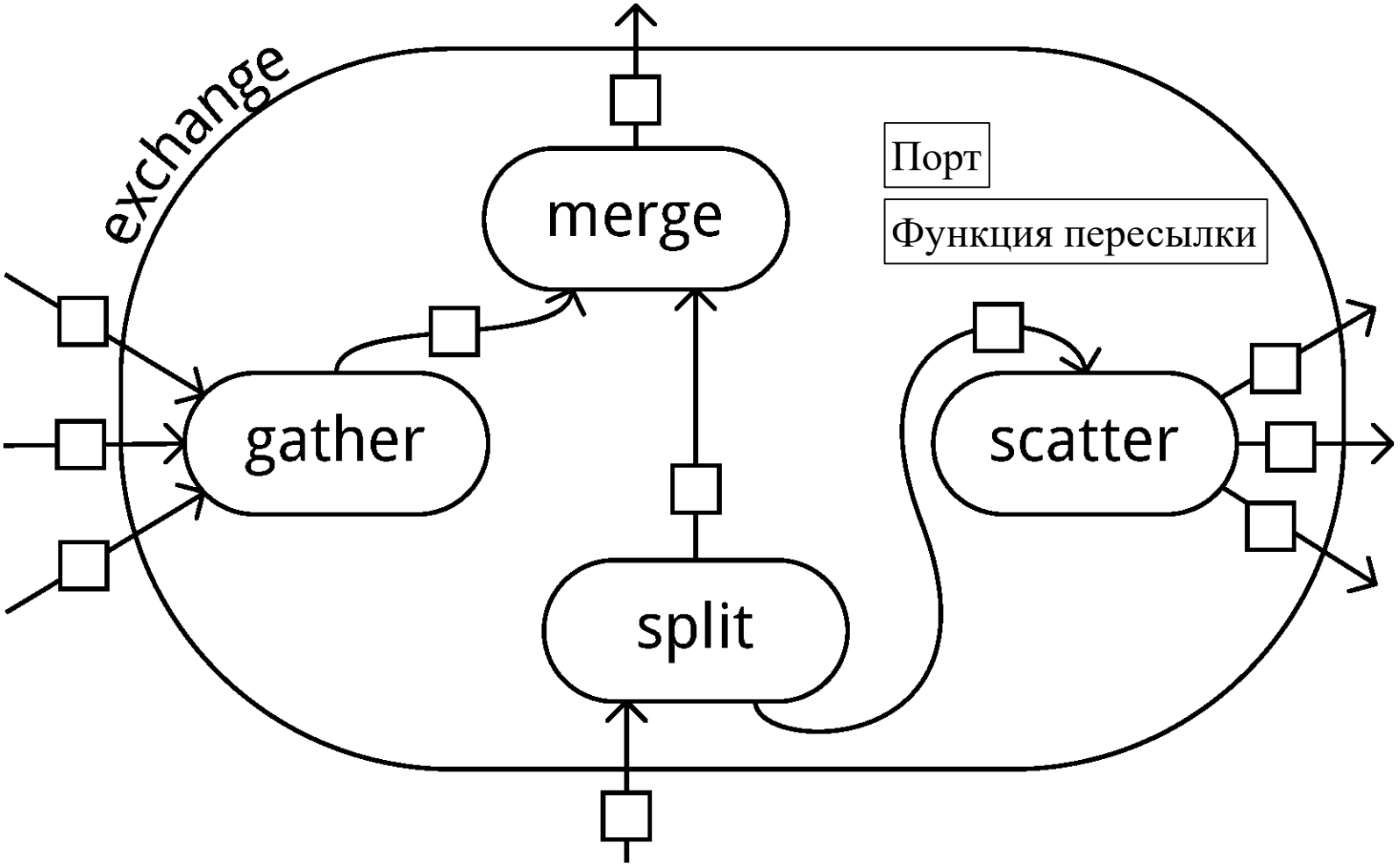
Передача данных

Создание подчиненного процесса

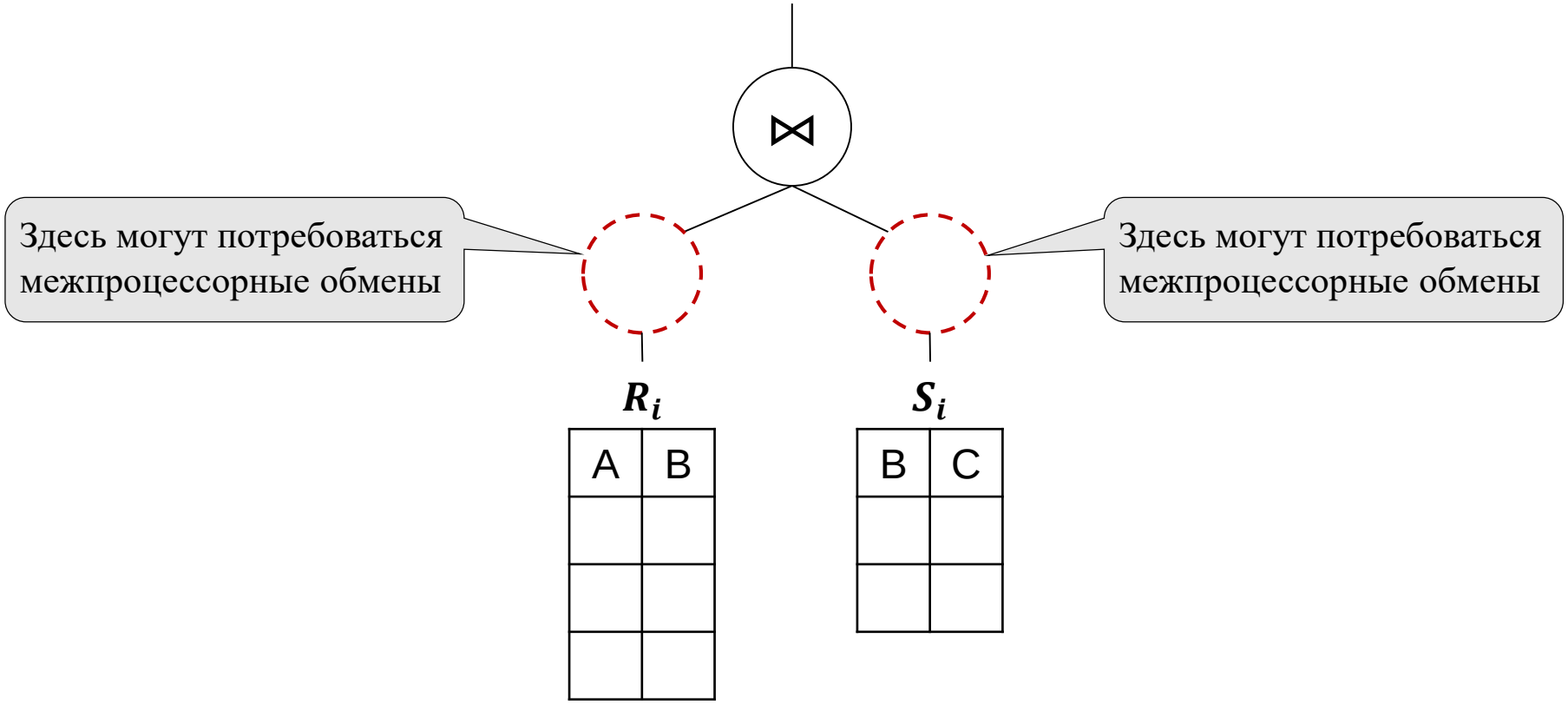
Процессы PargreSQL



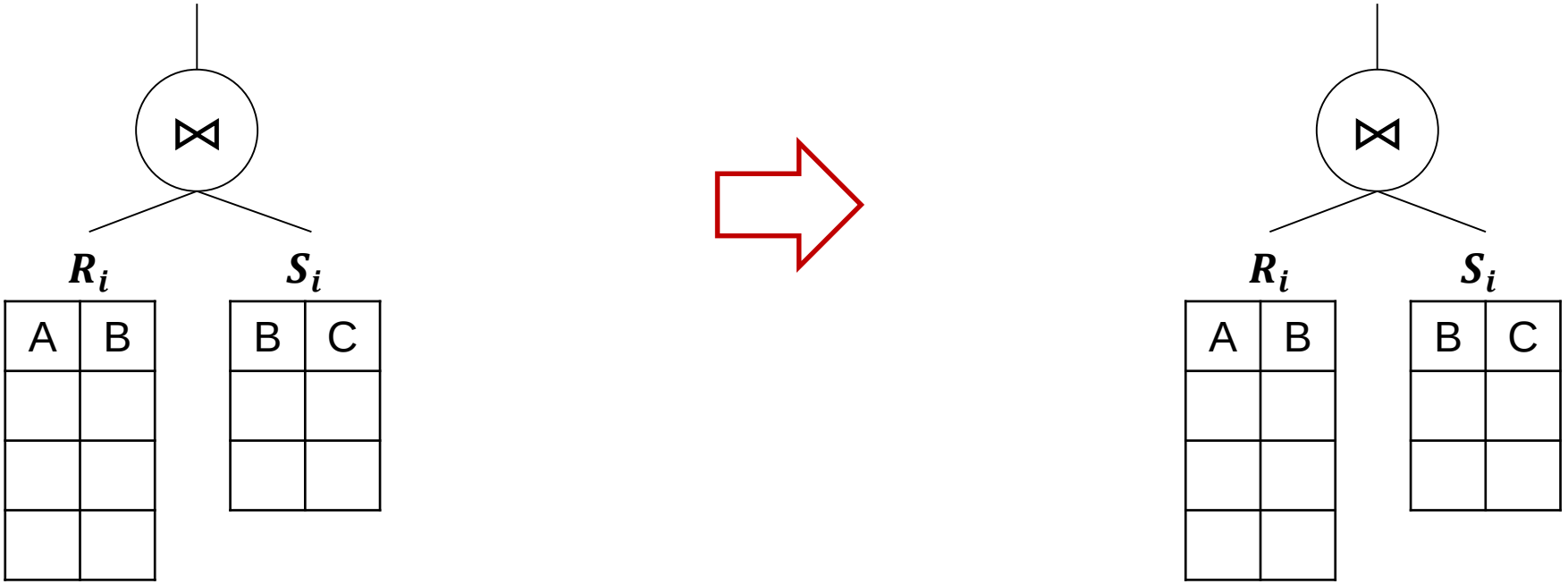
Оператор обмена EXCHANGE



Вставка операторов обмена на примере естественного соединения

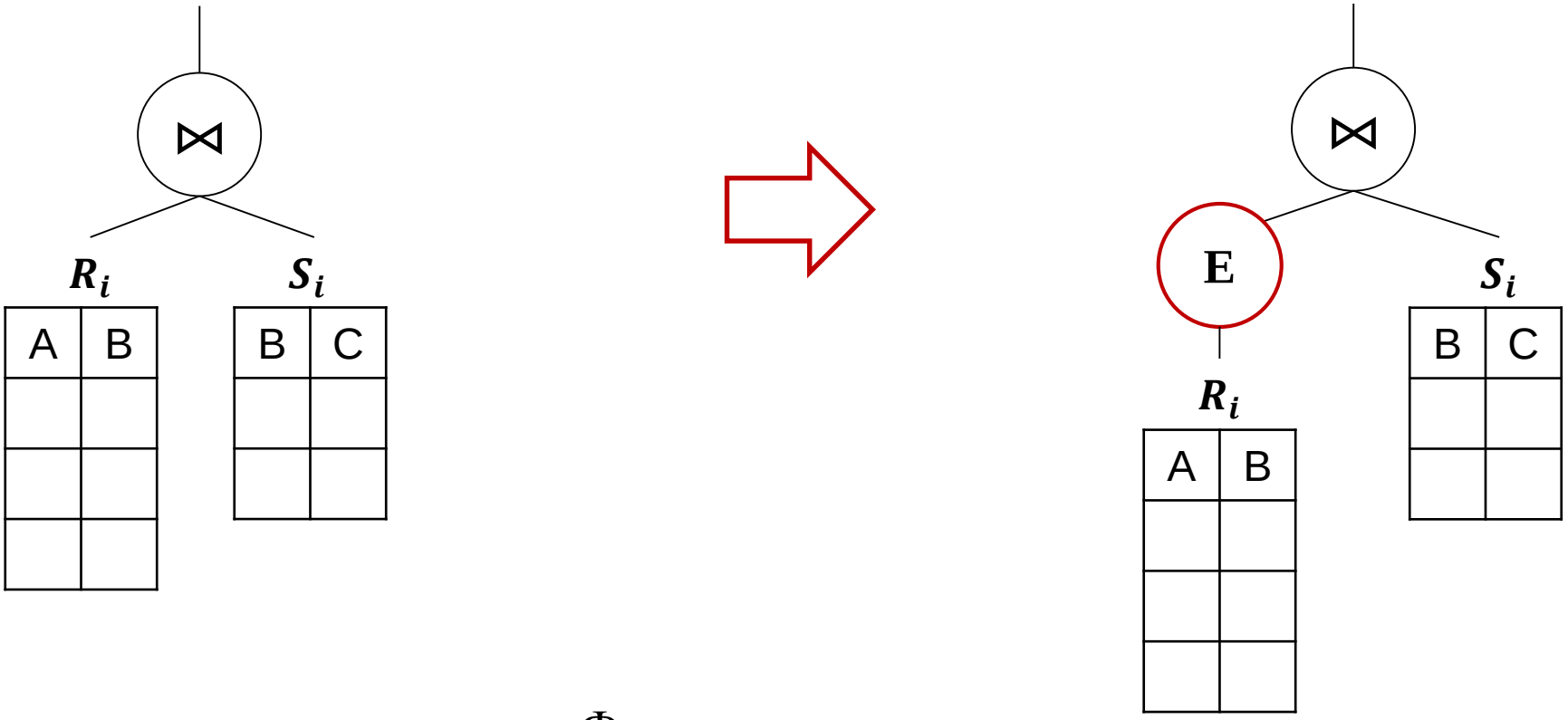


Случай 1: R и S фрагментированы по общему атрибуту с помощью одинаковой функции фрагментации



Обмены не нужны!

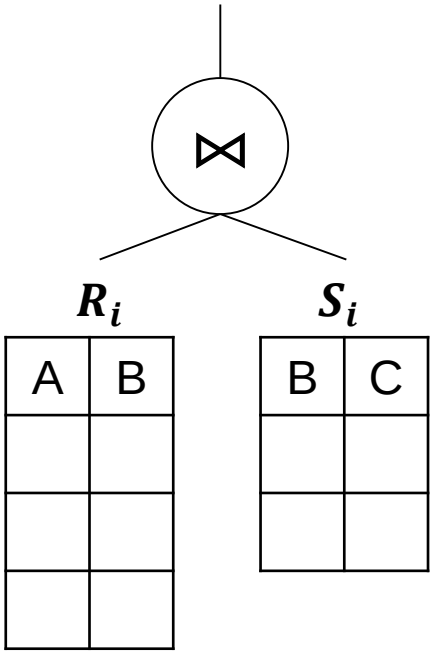
Случай 2: R фрагментировано по A, S фрагментировано по B с помощью функции φ_S

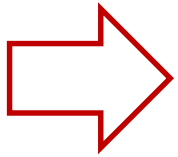


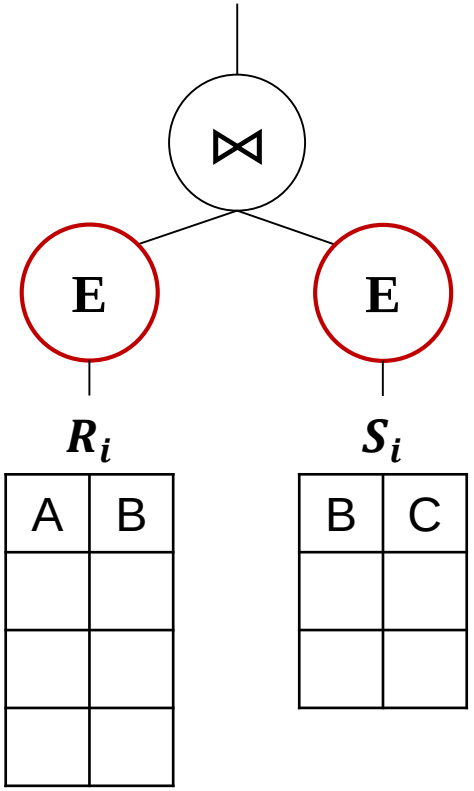
Функция пересылки:

$$\psi_R(r.B) = \varphi_S(r.B)$$

Случай 3: R фрагментировано по A, S фрагментировано по C





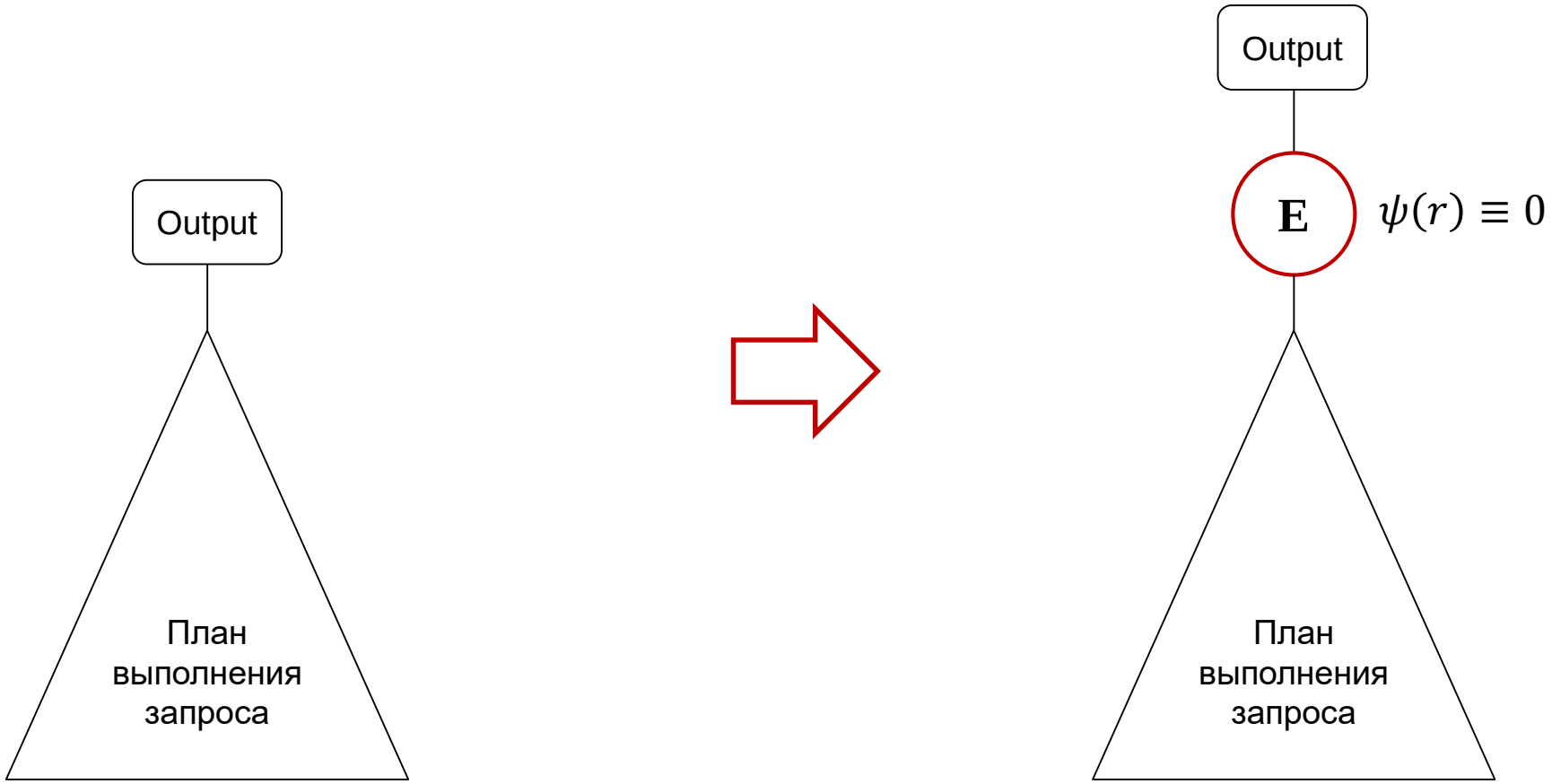


Функции пересылки:

$$\psi_R(r.B) = f(r.B)$$
$$\psi_S(s.B) = f(s.B)$$

$$f: D_B \rightarrow \{0, \ldots, P-1\}$$

Слияние результатов на сервере с номером 0



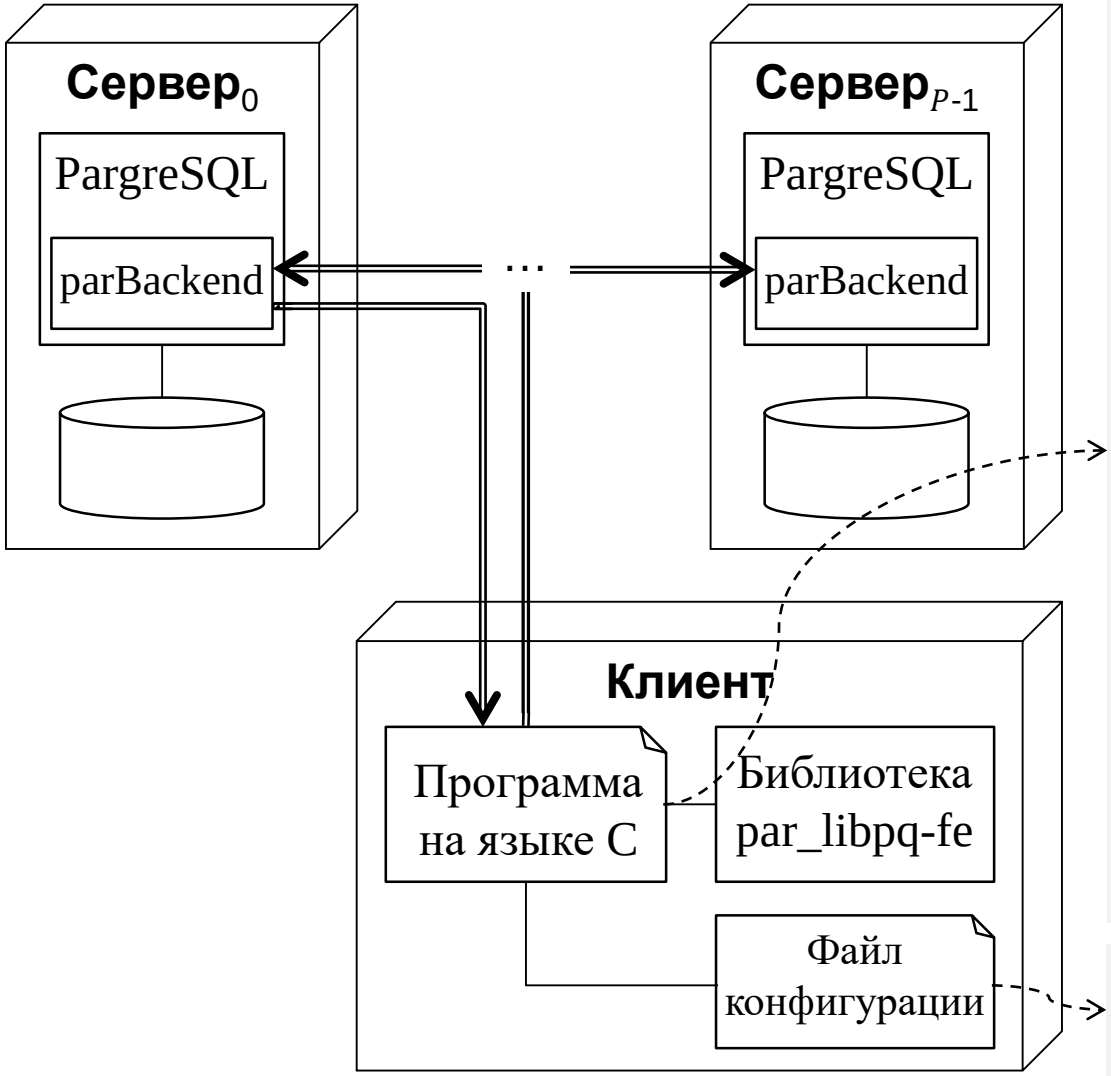
Реализация фрагментации

```
CREATE TABLE R (A int, B int)
WITH (FRAGATTR = B);
```

Функция фрагментации по атрибуту B:

$$\varphi(r) = r.B \ mod \ P$$

Цикл обработки запросов в системе PargreSQL



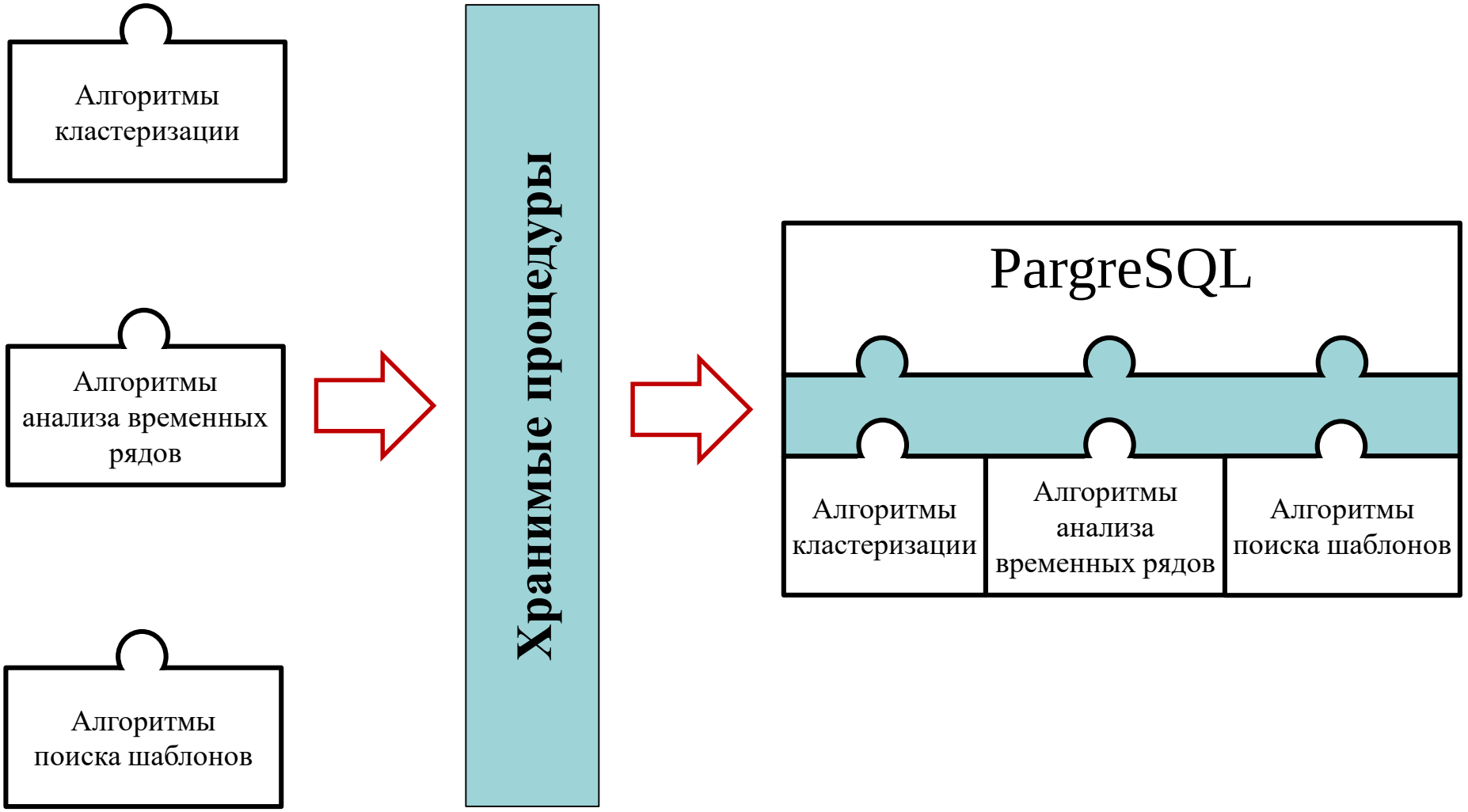

```
#include <par_libpq-fe.h>

PGconn *conn;
char *conninfo = "dbname=… hostaddr=… port=… ";
PGresult *res;

void main(int argc, char * argv[]) {
  conn = PQconnectdb (conninfo);

  res = PQexec(conn,
    "CREATE TABLE R (A INT, B INT)
  WITH (FRAGATTR = B);");

  res = PQexec(conn,
    "INSERT INTO R VALUES (2501, 1755);
    INSERT INTO R VALUES (2, 4);
    INSERT INTO R VALUES (8, 1); ");

  res = PQexec(conn, "SELECT * FROM R;");
  PQprint(stderr, res);

  PQclear(res);
  PQfinish(conn);
}
```

```
# Строки подключения к экземплярам PargreSQL
dbname=postgres hostaddr=10.4.5.204 port=5432
dbname=postgres hostaddr=10.4.5.205 port=5432
```

Использование PargreSQL

Предложенные методы используются
в компании Postgres Professional (Москва)
для разработки параллельной версии
СУБД PostgreSQL (проект ShardMan
https://github.com/postgrespro/shardman/)

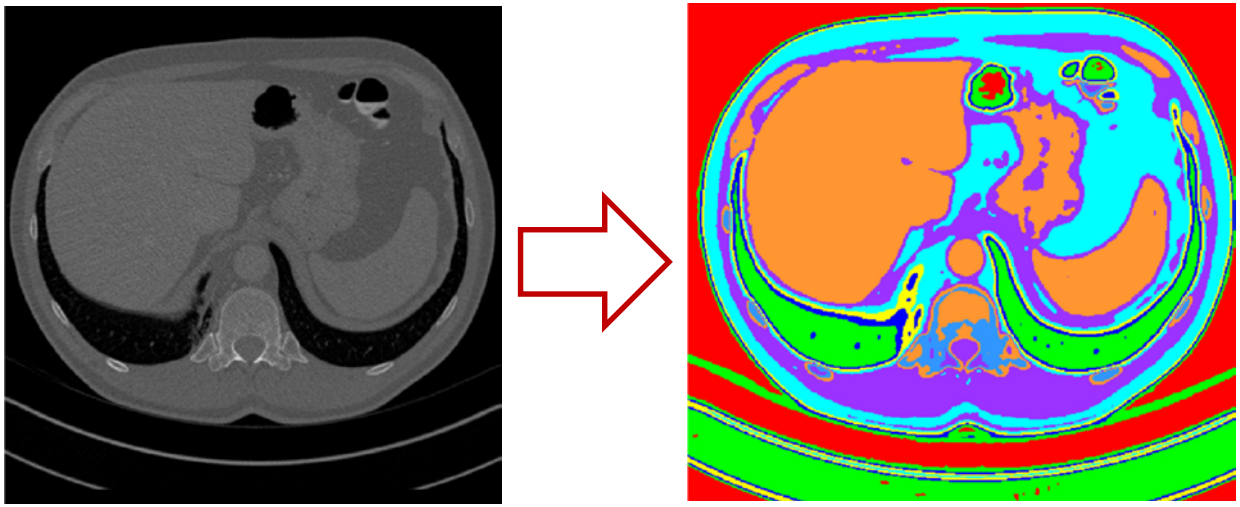
Внедрение алгоритмов анализа данных в PargreSQL



Комплекс алгоритмов интеллектуального анализа данных средствами параллельной СУБД

1. Алгоритмы кластеризации
2. Алгоритмы анализа временных рядов
3. Алгоритмы поиска шаблонов

Задача кластеризации

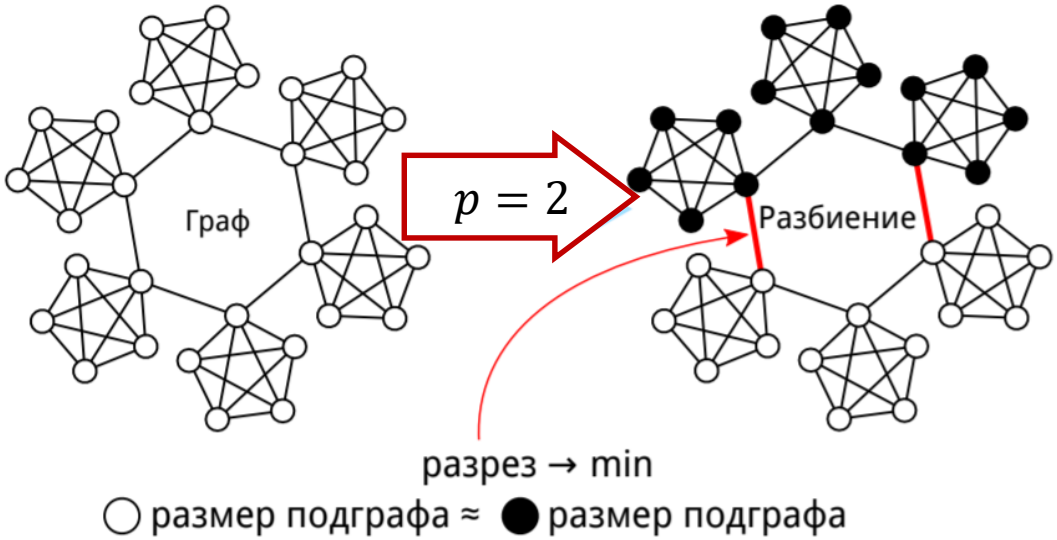


Множество объектов	$X = \{x_i \mid 1 \le i \le n, x_i \in \mathbb{R}^d\}$
Функция расстояния	$\rho(x_i, x_j)$
Количество кластеров	$k, 1 < k \ll n$
Метки кластеров	$C = \{c_i \mid 1 \le i \le k, c_i \in \mathbb{N}\}$
Алгоритм кластеризации	$\alpha: X \to C$

Алгоритмы кластеризации

- Кластеризация графа
- Нечеткая кластеризация данных
- Кластеризация на основе техники медоидов

Кластеризация графа



Граф $G(N, E, w)$

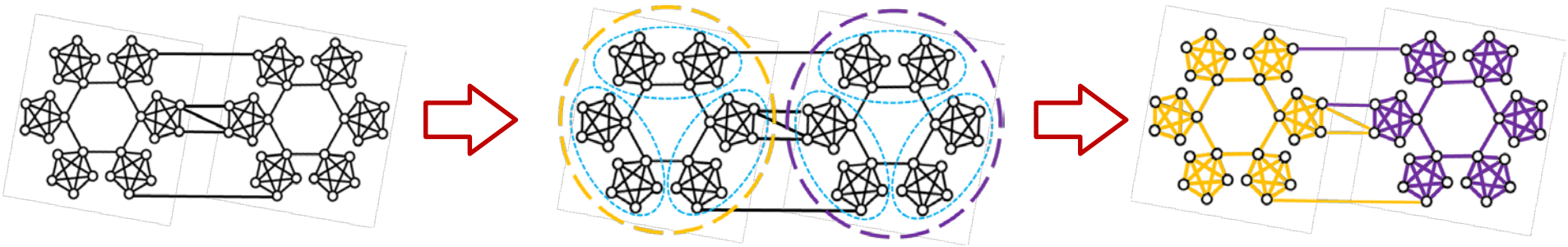
1. $N = \cup_{i=1}^{p} N_i,\ \forall i \neq j\ N_i \cap N_j = \emptyset,\ p > 1$
2. $w(N_i) \approx \frac{w(N)}{p}\ \ \forall i \in \{1, \dots, p\}$
3. *Разрез* $W_{cut} \to \min$, $W_{cut} := \sum_{e \in E_{cut}} w(e)$,
   $E_{cut} := \{(u, v) \in E \mid u \in N_i, v \in N_j, 1 \leq i, j \leq p, i \neq j\}$

Алгоритм dbParGraph

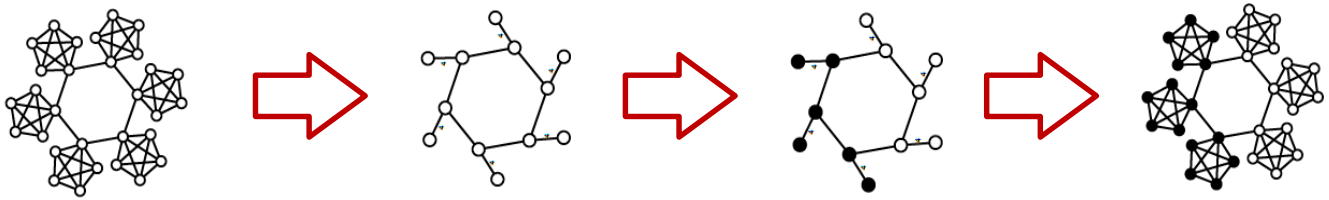
- Область применения
  - Кластеризация сверхбольших социальных графов
- Платформа
  - Параллельная СУБД для кластерной системы
- Отличительные особенности
  - Выявление сообществ в социальном графе

Структура алгоритма dbParGraph

1. Поиск и консолидация сообществ

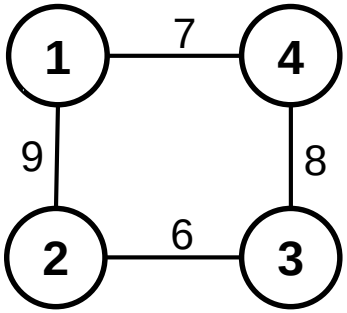


2. Многоуровневое разбиение



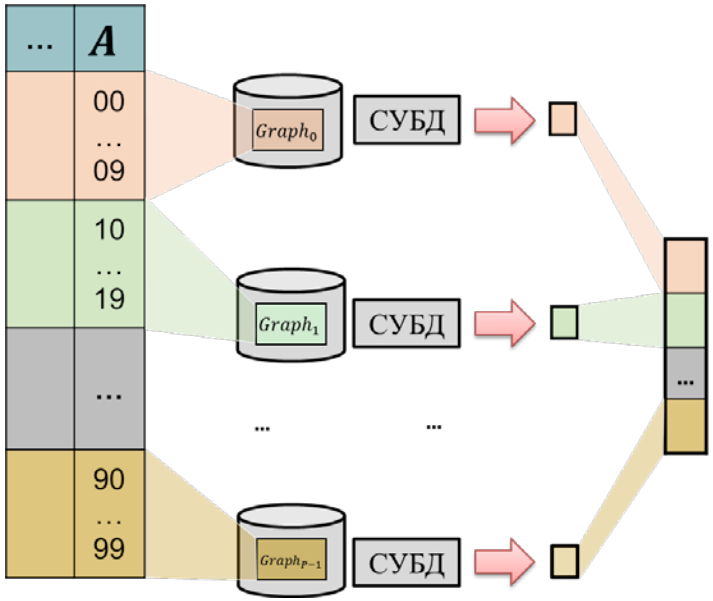
Реляционное представление графа

Исходный граф



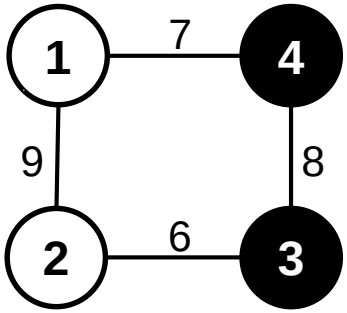
Отношение G (список ребер)

A	B	W
1	2	9
2	3	6
3	4	8
4	1	7



G фрагментировано по атрибуту A:

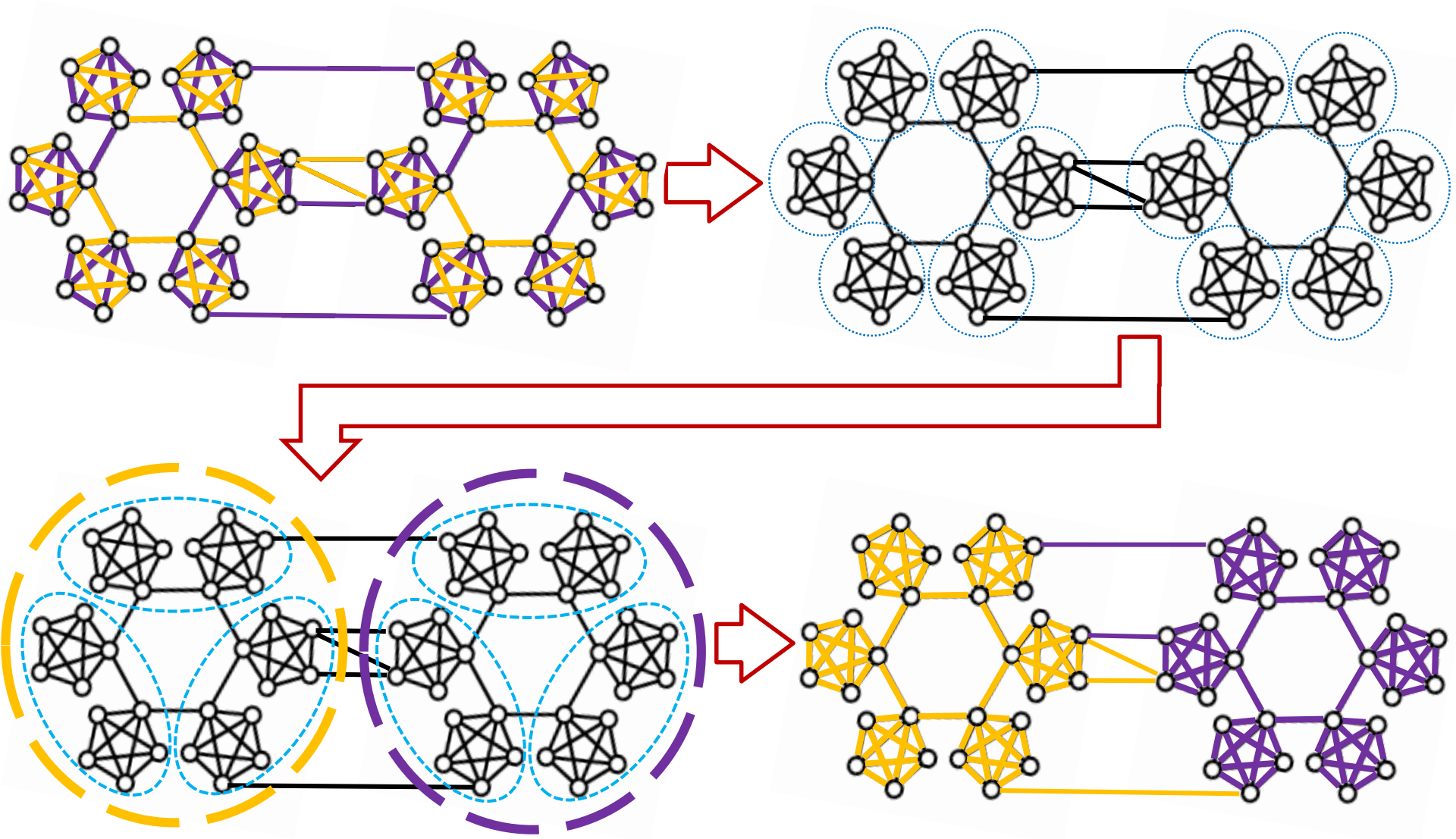
$$\varphi(g) = g.A \bmod P$$



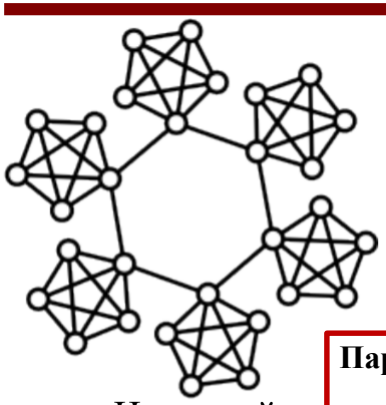
Отношение $Partitions$ (список вершин с метками кластеров)

A	P
1	**1**
2	**1**
3	**0**
4	**0**

Поиск и консолидация сообществ в графе

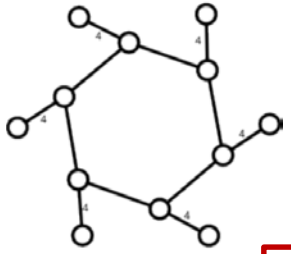


Многоуровневое разбиение



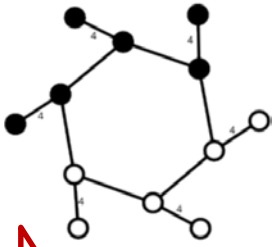
Исходный граф

Параллельная СУБД
Огрубление



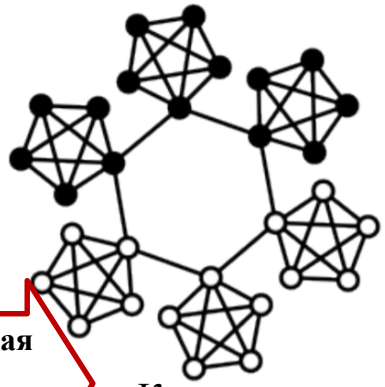
Огрубленный граф

Алгоритм в оперативной памяти
Нач. разбиение



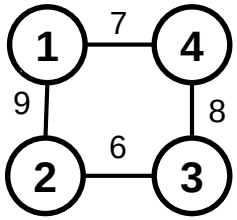
Грубое разбиение

Параллельная СУБД
Уточнение

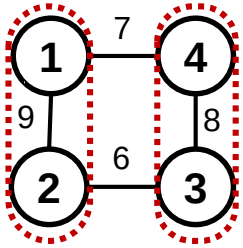


Кластери-зованный граф

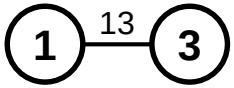
A	B	W
1	2	9
2	3	6
3	4	8
4	1	7



A	B
1	2
3	4



A	B	W
1	3	13

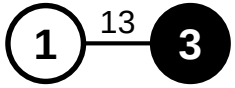


$$g(v) := ext(v) - int(v)$$

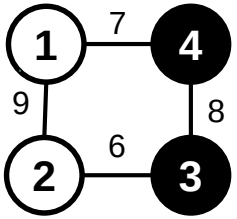
$$ext(v) := \sum_{(v,u)\in E, P(v)\neq P(u)} w(v,u)$$

$$int(v) := \sum_{(v,u)\in E, P(v)=P(u)} w(v,u)$$

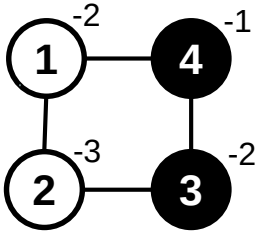
A	P
1	1
3	0



A	P
1	1
2	1
3	0
4	0



A	P	G
1	1	-2
2	1	-3
3	0	-2
4	0	-1



Сравнение с аналогами

Алгоритм	Граф		Время (сек.)	***dbPar graph*** (сек.)
	$\|N\|$	$\|E\|$		
MSP [1]	10^7	$5 \cdot 10^7$	1 305	**1 189**
ParMETIS [2]	$7.7 \cdot 10^6$	$1.33 \cdot 10^8$	1 004	**886**
PT-Scotch [3]	$2.3 \cdot 10^7$	$1.75 \cdot 10^8$	1 148	**897**

[1] Zeng Z., *et al.* A parallel graph partitioning algorithm to speed up the large-scale distributed graph mining. BigMine 2012. pp. 61–68.

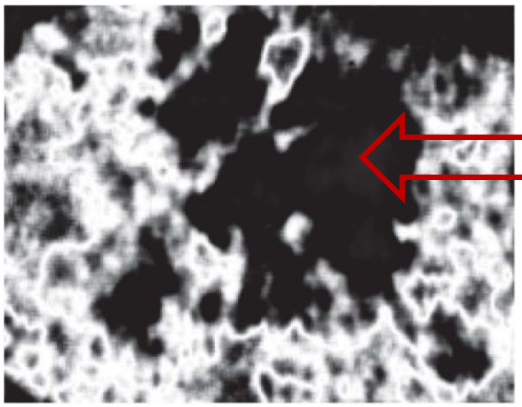
[2] Karypis G. METIS and ParMETIS. Enc. of Parallel Computing (Ed. by D.A. Padua). Springer, 2011. pp. 1117–1124.

[3] Chevalier C., Pellegrini F. PT-Scotch: A tool for efficient parallel graph ordering. Parallel Computing. 2008. vol. 34, no. 6-8. pp. 318–331.

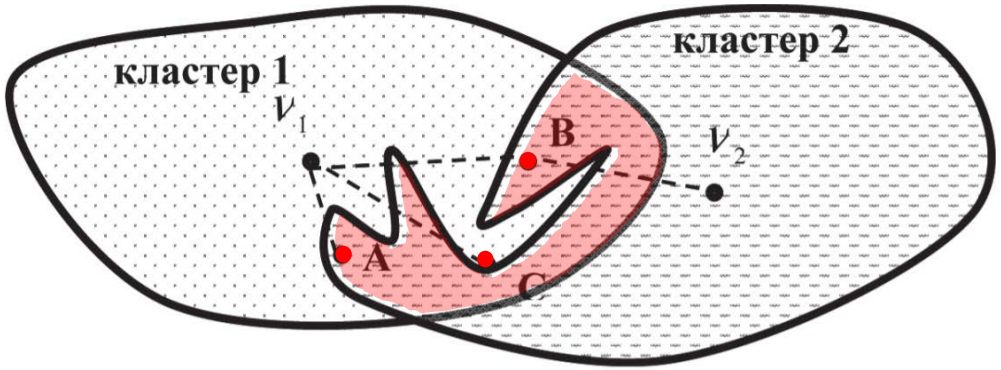
Алгоритмы кластеризации

- Кластеризация графа
- **Нечеткая кластеризация данных**
- Кластеризация на основе техники медоидов

Нечеткая кластеризация данных



Радиологическое изображение



Сегментация радиологического изображения

- $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ – матрица объектов
- $k, 1 < k \ll n$ – количество кластеров
- $\boldsymbol{C} \in \mathbb{R}^{k \times d}$ – матрица центроидов
- $\boldsymbol{U} \in \mathbb{R}^{n \times k}$ – матрица принадлежности объектов кластерам:

$$0 \le u_{ij} \le 1, \sum_{j=1}^{k} u_{ij} = 1$$

- ρ – функция евклидова расстояния
- Целевая функция:

$$J_{\text{FCM}}(\boldsymbol{X}, k, m) = \sum_{j=1}^{k} u_{ij}^{m} \rho^2(x_i, c_j)$$

- $m > 1$ – степень нечеткости (параметр)

$\boldsymbol{U}^{(0)} := random(0..1); s := 0$

повторять

Вычислить матрицу центроидов

$$\boldsymbol{C}^{(s)} := (c_{jl}),\ c_{jl} := \frac{\sum_{i=1}^{n} u_{ij}^{m} \cdot x_{il}}{\sum_{i=1}^{n} u_{ij}^{m}}$$

Вычислить матрицу принадлежности

$$\boldsymbol{U}^{(s+1)} := (u_{ij}),\ u_{ij} := \sum_{t=1}^{k} \left(\frac{\rho(x_i, c_j)}{\rho(x_i, c_t)}\right)^{\frac{2}{1-m}}$$

$s := s + 1$

пока $\max_{ij} \left\{ \left| u_{ij}^{(s)} - u_{ij}^{(s-1)} \right| \right\} \le \varepsilon$

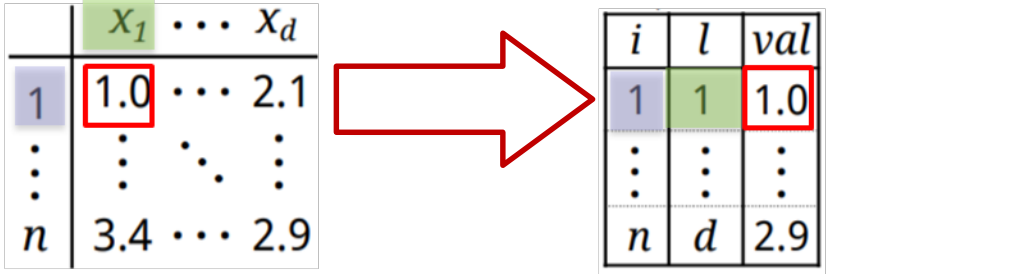
Алгоритм pgFCM

- Область применения
  - Нечеткая кластеризация сверхбольших данных
- Платформа
  - Параллельная СУБД для кластерной системы

Структура алгоритма pgFCM

1. «Индексное» представление данных

Матрица объектов $\boldsymbol{X}$ Таблица объектов $\boldsymbol{SV}$



2. Использование высокоуровневых SQL-запросов

```
CREATE TEMP TABLE U (
    i int, j int, val numeric,
    PRIMARY KEY (i,j));

CREATE TEMP TABLE P (
    d int, k int, n int, s int, delta numeric,
    PRIMARY KEY (d,k,n));

CREATE TEMP TABLE SV (
    i int, l int, val numeric,
    PRIMARY KEY (i,l));
```

```
-- Вычисление центров кластеров
INSERT INTO C
    SELECT R.j, SV.l, sum(R.s * SV.val) / sum(R.s) AS val
    FROM (
        SELECT i, j, U.val^m AS s
        FROM U) AS R, SV
    WHERE R.i = SV.i
    GROUP BY j, l;

-- Вычисление расстояний
INSERT INTO SD
    SELECT i, j, sqrt(sum((SV.val - C.val)^2)) as dist
    FROM SV, C
    WHERE SV.l = C.l;
    GROUP BY i, j;
```

Индексное представление данных

Матрица объектов $\boldsymbol{X}$

	x_1	$\cdots$	x_d
1	1.0	$\cdots$	2.1
$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	3.4	$\cdots$	2.9

Таблица объектов $\boldsymbol{SV}$

i	l	val
1	1	1.0
$\vdots$	$\vdots$	$\vdots$
n	d	2.9

Таблица объектов $\boldsymbol{SH}$

i	x_1	$\cdots$	x_d
1	1.0	$\cdots$	2.1
$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	3.4	$\cdots$	2.9

Матрица центроидов $\boldsymbol{C}$

	x_1	$\cdots$	x_d
1	2.2	$\cdots$	8.1
$\vdots$	$\vdots$	$\ddots$	$\vdots$
k	3.4	$\cdots$	6.9

Таблица центроидов $\boldsymbol{C}$

j	l	val
1	1	2.2
$\vdots$	$\vdots$	$\vdots$
k	d	6.9

Матрица принадлежности $\boldsymbol{U}$

	1	$\cdots$	k
1	0.2	$\cdots$	0.1
$\vdots$	$\vdots$	$\ddots$	$\vdots$
n	0.8	$\cdots$	0.1

Таблица $\boldsymbol{U}$ (шаг s)

i	j	val
1	1	0.2
$\vdots$	$\vdots$	$\vdots$
n	k	0.1

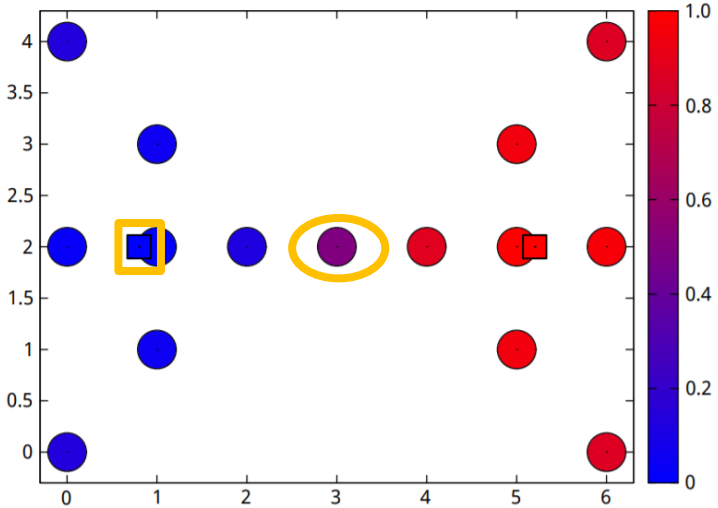
Таблица $\boldsymbol{UT}$ (шаг $s+1$)

i	j	val
1	1	0.2
$\vdots$	$\vdots$	$\vdots$
n	k	0.1

Использование высокоуровневых SQL-запросов

Алгоритм PGFCM(**вход**: таблица $\boldsymbol{SH}, m, \varepsilon, k$; **выход**: таблицы $\boldsymbol{U}, \boldsymbol{C}$)
Создать таблицы $\boldsymbol{U}$, $\boldsymbol{P}$, $\boldsymbol{SV}$ и др.
повторять
- Вычислить центроиды c_j
- Обновить таблицу центроидов $\boldsymbol{C}$
- Вычислить расстояния $\rho(x_i, c_j)$
- Обновить таблицу $\boldsymbol{SD}$
- Вычислить ut_{ij}
- Обновить таблицу $\boldsymbol{UT}$
- Обновить таблицы $\boldsymbol{U}$, $\boldsymbol{P}$

пока $\boldsymbol{P}.delta > \varepsilon$
выдать таблицы $\boldsymbol{U}, \boldsymbol{C}$



$k = 2$

Таблица $\boldsymbol{U}$

i	j	val
1	1	0.86
1	2	0.13
2	1	0.97
2	2	0.02
...	...	...
8	1	0.49
8	2	0.50
...	...	...
15	1	0.13
15	2	0.86

Таблица $\boldsymbol{SH}$

i	x_1	x_2
1	0	0
2	0	2
3	0	4
8	3	2
	...	
15	6	0

Таблица $\boldsymbol{C}$

j	l	val
1	1	0.79
1	2	2.0
2	1	5.2
2	2	1.99

Сравнение с аналогами

Алгоритм	Данные		Время (сек.)	***pgFCM*** (сек.)
	n	d		
WCFC [1)]	$4.9 \cdot 10^6$	41	5 313	**5 182**
BigFCM [2)]	$1.1 \cdot 10^7$	28	611	**531**

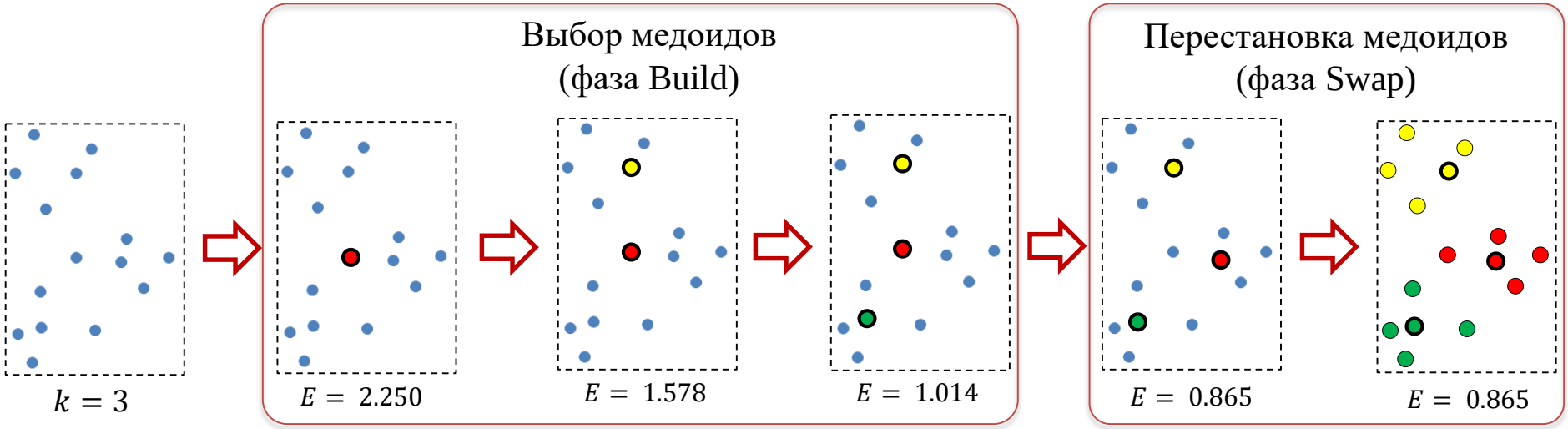
[1)] Hidri M.S., *et al.* Speeding up the large-scale consensus fuzzy clustering for handling Big Data. Fuzzy Sets and Systems. 2018. vol. 348. pp. 50–74.

[2)] Ghadiri N., *et al*. BigFCM: Fast, precise and scalable FCM on Hadoop. Future Generation Comp. Syst. 2017. vol. 77. pp. 29–39.

Алгоритмы кластеризации

- Кластеризация графа
- Нечеткая кластеризация данных
- **Кластеризация на основе техники медоидов**

Кластеризация данных на основе техники медоидов



Медоид кластера c_i $\quad med_{c_i} := \arg\min_{x_j \in c_i} \rho\,(x_j, r_{c_i})$, где r_{c_i} – центр кластера c_i

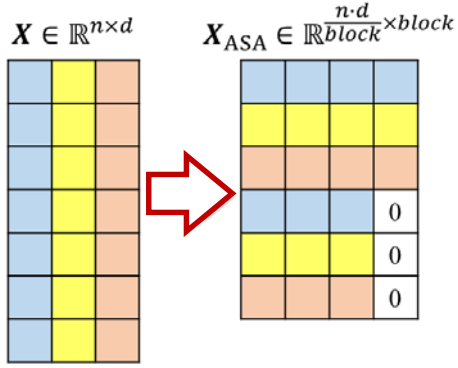
Целевая функция $E: X \times X \to \mathbb{R}$ $\quad E := \Sigma_{j=1}^{n} \min_{1 \le i \le k} \rho(med_{c_i}, x_j)$

Алгоритм PPAM

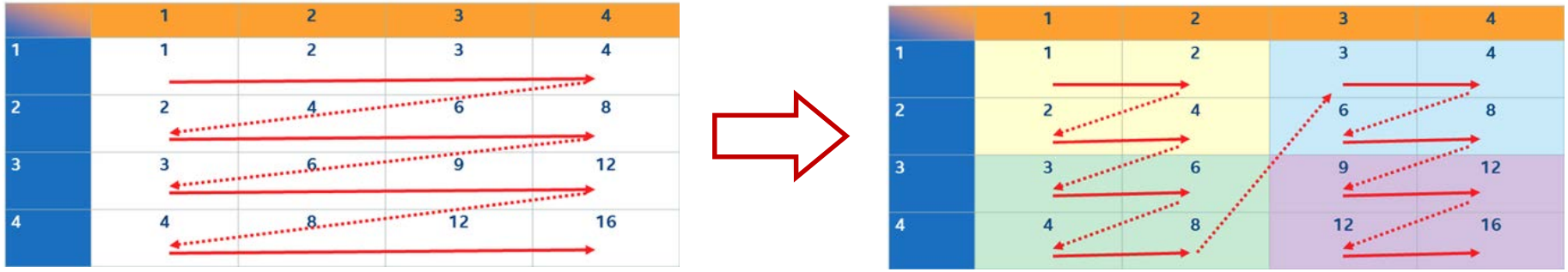
- Область применения
  - Кластеризация данных с выбросами и шумами
- Платформа
  - Многоядерные ускорители
- Отличительные особенности
  - Предвычисление матрицы расстояний
  - Эффективное использование векторизации

Структура алгоритма PPAM

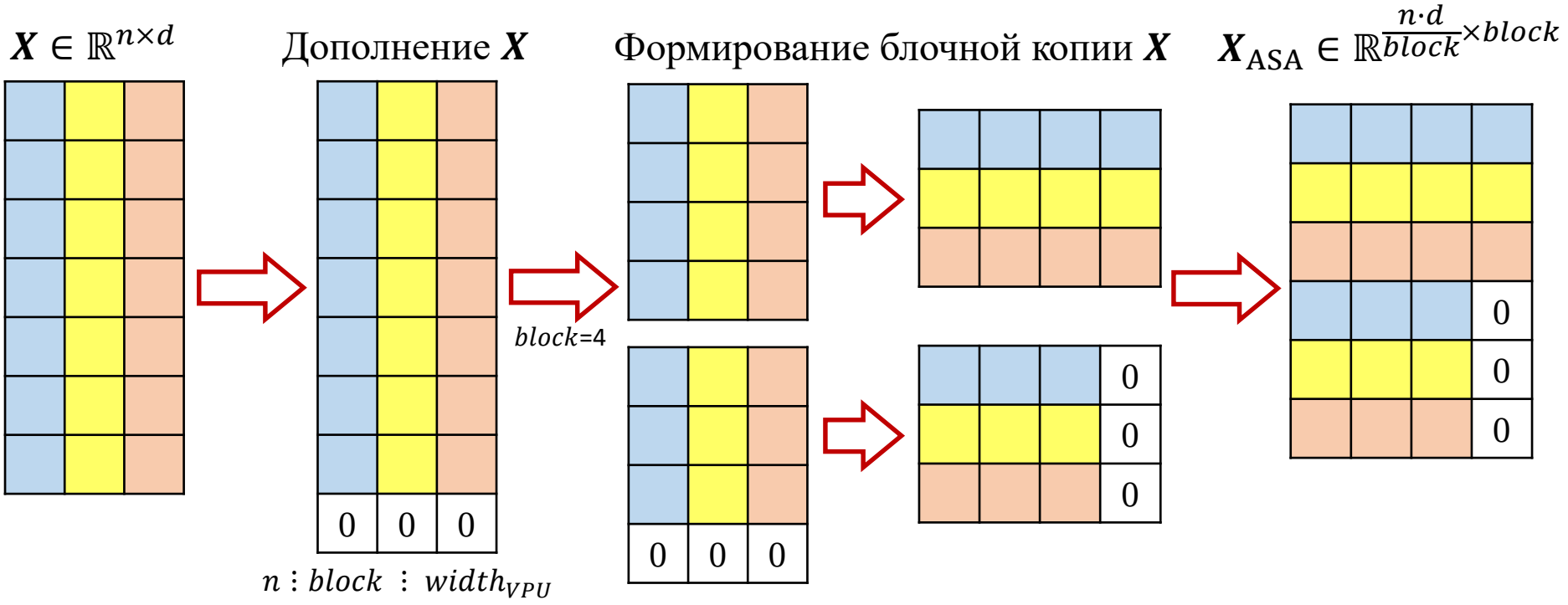
1. Предвычисление матрицы расстояний
   с помощью блочной копии матрицы объектов



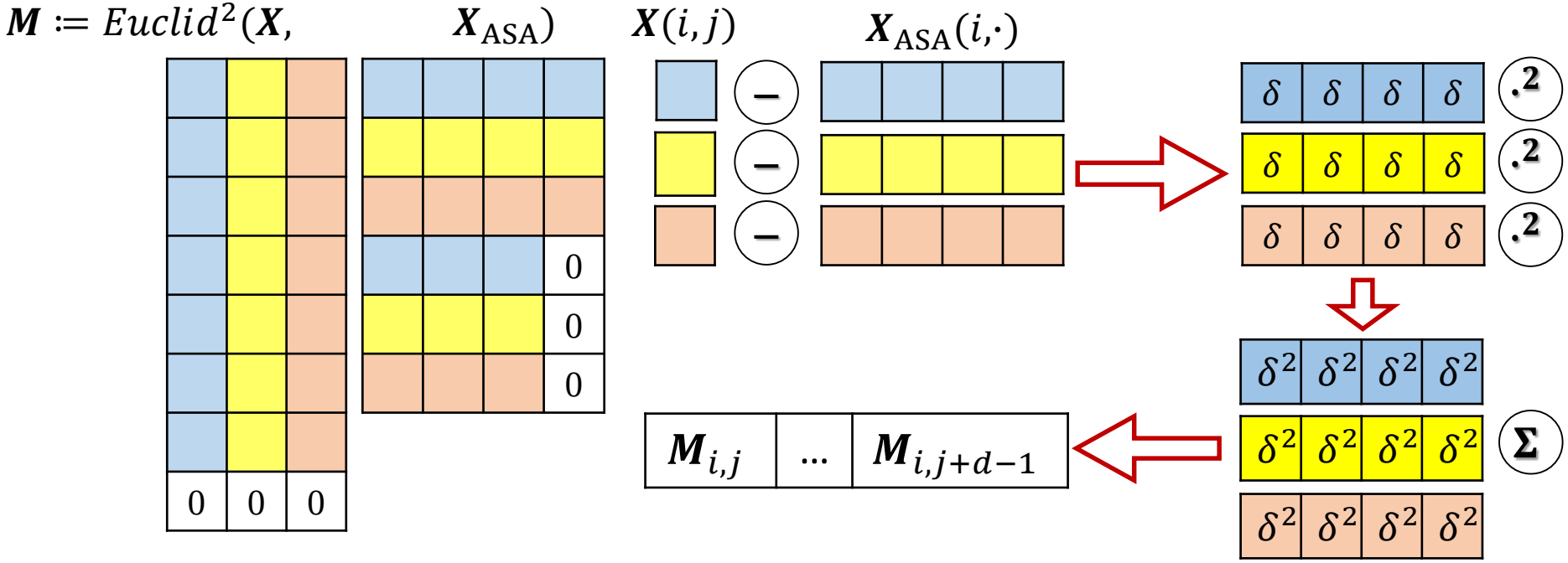
2. Выбор и перестановка медоидов
   с помощью тайлинга циклов



Создание блочной копии матрицы объектов



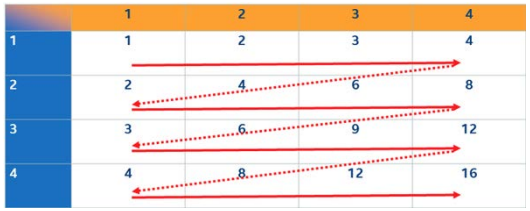
Предвычисление матрицы расстояний. Выбор и перестановка с использованием тайлинга

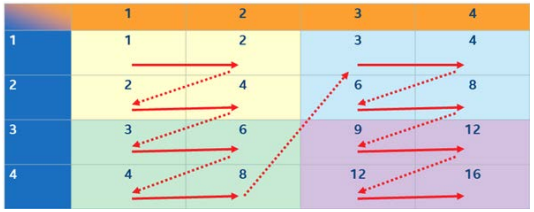


parallel for $i := 1$ **to** n **do**
$Process(M(i,j))$

Тайлинг

parallel for $i := 1$ **to** n **step** $tile$ **do**
for $i_{tile} := i$ **to** $i + tile$ **do**
$Process(M(i_{tile}, j))$





Сравнение с аналогами

Алгоритм	Время/Качество кластеризации	
	Набор RSSI [1] ($d = 15, n = 6.5 \cdot 10^3$)	Набор Avila [2] ($d = 10, n = 2.1 \cdot 10^4$)
k-Medians [3]	81.1	163.2
k-Medoids [4]	83.3	52.7
PPAM	**38.3**	**42.2**

[1] Mohammadi M., Al-Fuqaha A.I. Enabling cognitive Smart Cities using Big Data and Machine Learning: approaches and challenges. IEEE Comm. Magazine. 2018. vol. 56, no. 2. pp. 94–101.

[2] De Stefano C., *et al.* Reliable writer identification in medieval manuscripts through page layout features: The "Avila" Bible case. Engineering Applications of Artificial Intelligence. 2018. vol. 72. pp. 99–110.

[3] Guha S., *et al.* Clustering data streams. FOCS 2000. pp. 359–366.

[4] Ng R.T., Han J. Efficient and effective clustering methods for spatial Data Mining. VLDB'94. pp. 144–155.

Комплекс алгоритмов интеллектуального анализа данных средствами параллельной СУБД

1. Алгоритмы кластеризации
2. **Алгоритмы анализа временных рядов**
3. Алгоритмы поиска шаблонов

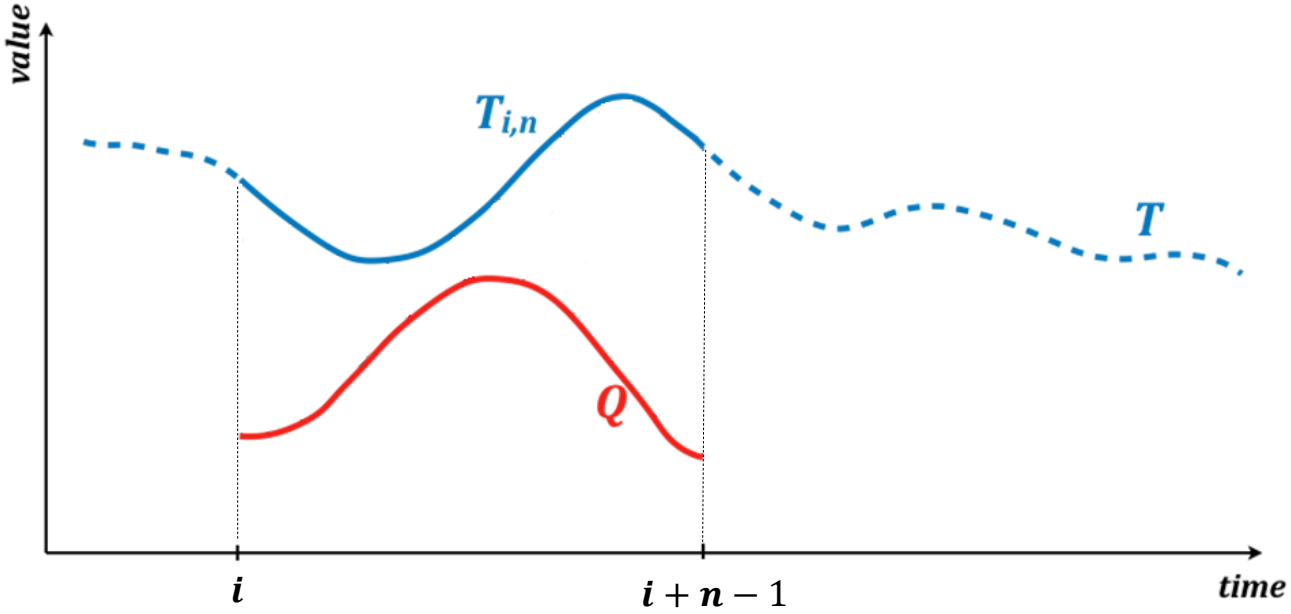
Анализ временных рядов

Временной ряд – упорядоченная последовательность числовых значений с отметками времени

Основные задачи

- Поиск похожих подпоследовательностей
- Поиск аномальных подпоследовательностей

Поиск наиболее похожей подпоследовательности



- Временной ряд: $T := (t_1, \dots, t_m), t_i \in \mathbb{R}$
- Подпоследовательность: $T_{i,n} := (t_i, \dots, t_{i+n-1}), 1 \le i \le m-n+1, n \ll m$
- Образец поиска: $Q = (q_1, \dots, q_n)$
- Мера схожести подпоследовательностей: DTW
- Подпоследовательность $T_{i,n}$ наиболее похожа на образец поиска Q, если

$$\forall\, T_{j,n} \;\; DTW\left(Q, T_{i,n}\right) \le DTW\left(Q, T_{j,n}\right)$$

Мера схожести
DTW (Dynamic Time Warping)

$$DTW(Q,C) \coloneqq d(n,n)$$

$$d(i,j) \coloneqq (q_i - c_j)^2 + \min \begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1) \end{cases}$$

$$d(0,0) = 0, \qquad d(i,0) = d(0,j) = +\infty$$

$$C = (4,3,3,4,5,4), Q = (1,2,3,3,2,1)$$

$r = 6$

		4	3	3	4	5	4	
1	+∞	28	15	15	20	30	**28**	6
2	+∞	19	11	11	14	20	19	5
3	+∞	15	10	10	11	15	16	4
3	+∞	14	10	10	11	15	16	3
2	+∞	13	10	11	15	24	28	2
1	+∞	9	13	17	26	42	51	1
	0	+∞	+∞	+∞	+∞	+∞	+∞	0
→ *j*	0	1	2	3	4	5	6	↑ *i*

$r = 3$

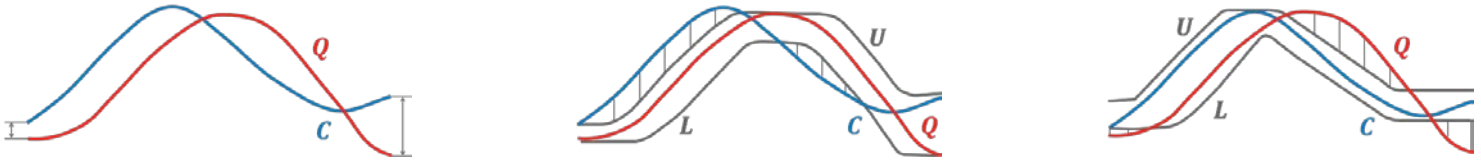
		4	3	3	4	5	4	
1	+∞	0	0	0	9	20	**14**	6
2	+∞	0	0	1	4	10	5	5
3	+∞	0	0	0	1	5	1	4
3	+∞	14	10	10	5	4	0	3
2	+∞	13	10	11	4	0	0	2
1	+∞	9	13	17	0	0	0	1
	0	+∞	+∞	+∞	+∞	+∞	+∞	0
→ *j*	0	1	2	3	4	5	6	↑ *i*

Алгоритм PBM

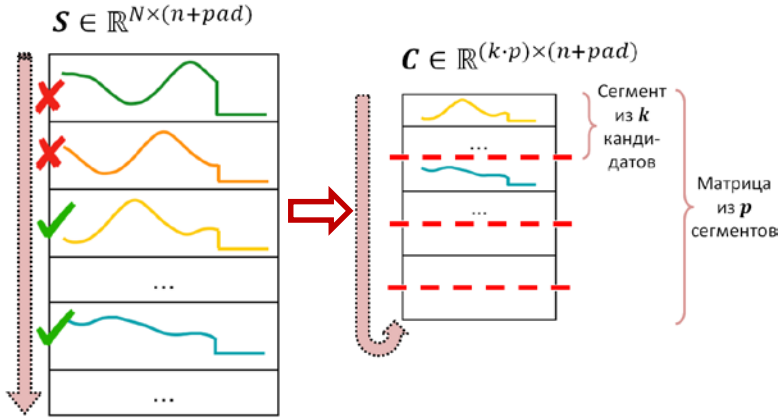
- Область применения
  - Поиск похожих подпоследовательностей в сверхбольших временных рядах
- Платформа
  - Кластерная система с многоядерными ускорителями
- Отличительные особенности
  - Масштабируемость, близкая к линейной
  - Эффективная векторизация вычислений

Структура алгоритма PBM

1. Предвычисление нижних границ схожести

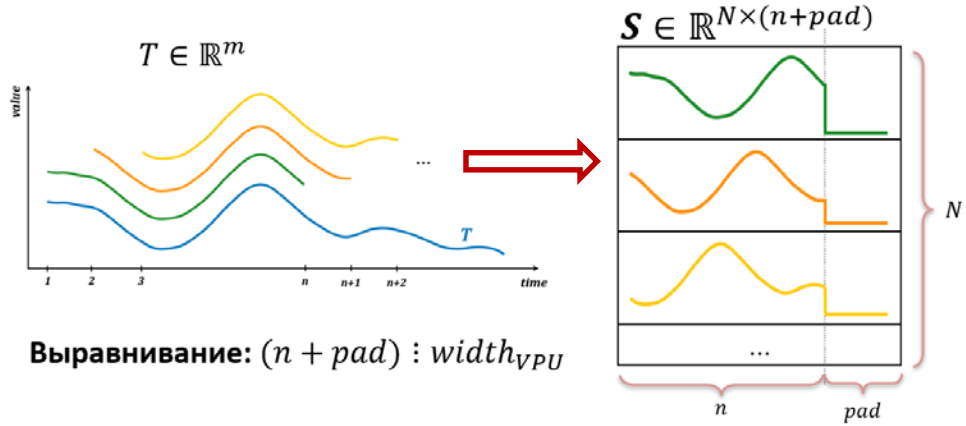


2. Улучшение порога схожести



Предвычисление нижних границ схожести

1. Сформировать S_T^n
2. **pardo** $\forall i \in 1..N$
3. $Normalize(S_T^n(i,\cdot))$
4. **do** $\forall k \in 1..lb_{max}$
5. $L_T^n(i,k) := LB_k(S_T^n(i,\cdot))$
6. **end do**
7. **end pardo**



Выравнивание: $(n + pad) \vdots width_{VPU}$

Матрица подпоследовательностей

$$\boldsymbol{S_T^n} \in \mathbb{R}^{N\times(n+pad)}$$

$$S_T^n(i,j) := t_{i+j-1}$$

$$N := |T| - n + 1$$

Нормализация

$$\hat{C} = (\hat{c}_1, \dots, \hat{c}_n), \hat{c}_i = \frac{c_i - \mu}{\sigma},$$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} c_i, \sigma^2 = \frac{1}{n}\sum_{i=1}^{n} {c^2}_i - \mu^2$$

Границы схожести

$$\boldsymbol{LB_1}(Q,C) := ED(\hat{q}_1, \hat{c}_1) + ED(\hat{q}_n, \hat{c}_n)$$

$$\boldsymbol{LB_2}(Q,C) := \sum_{i=1}^{n} \begin{cases} (\hat{c}_i - u_i)^2, if\ \hat{c}_i > u_i \\ (\hat{c}_i - \ell_i)^2, if\ \hat{c}_i < \ell_i \\ 0, \text{иначе} \end{cases}$$

$$u_i := \max_{i-r\le k\le i+r} \hat{q}_k, \qquad \ell_i := \min_{i-r\le k\le i+r} \hat{q}_k$$

$$\boldsymbol{LB_3}(Q,C) := LB_2(C,Q)$$

Матрица нижних границ

$$\boldsymbol{L_T^n} \in \mathbb{R}^{N\times lb_{max}}$$

Улучшение порога схожести

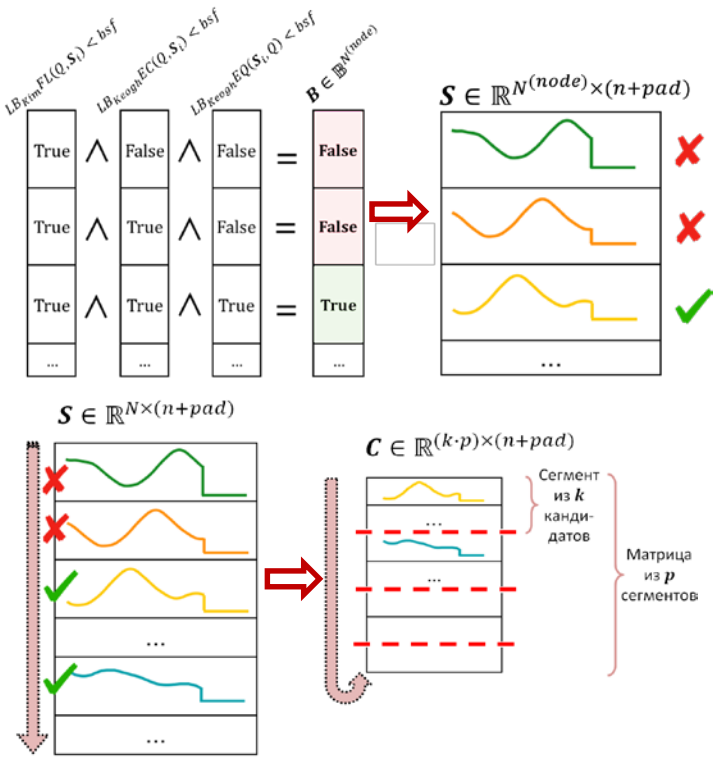
1. **pardo** $i \in 1..N$
2. $B_T^n(i) \coloneqq \bigwedge_{i=0}^{lb_{max}} (L_T^n(i,k) < bsf)$
3. **end pardo**
4. $C_T^n \coloneqq \{S_T^n(i,\cdot) \mid B_T^n(i) = \text{TRUE}\}$
5. **pardo** $c \in C_T^n$
6. $bsf \coloneqq \min_{0 \le i \le thread_{max}} DTW(c,Q)$
7. $position \coloneqq \arg\min_{0 \le i \le thread_{max}} DTW(c,Q)$
8. **end pardo**

Карта схожести

$$\boldsymbol{B_T^n} \in \mathbb{B}^N$$

Матрица кандидатов

$$\boldsymbol{C_T^n} \in \mathbb{R}^{(s \cdot thread_{max}) \times (n+pad)}$$



Сравнение с аналогами

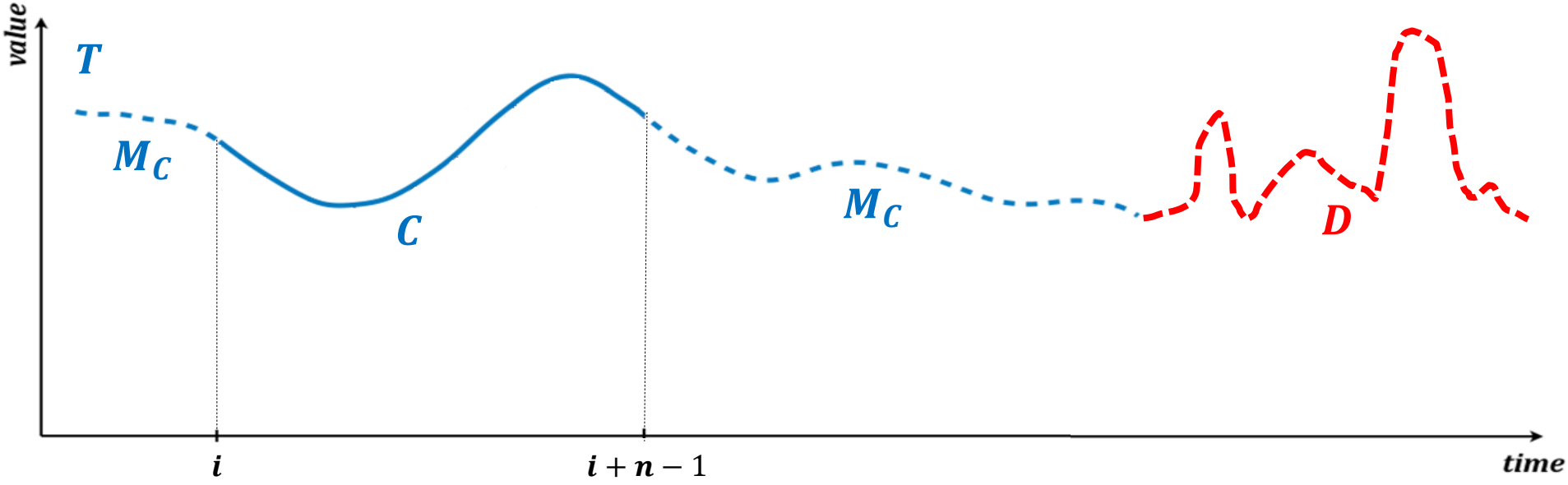
Алгоритм	Длина ряда	Время (сек.)	***PBM*** (сек.)
Shabib [1)]	$2.2 \cdot 10^8$	32	**24**
Sart [2)]	$1.5 \cdot 10^6$	80.4	**36.9**
Huang [3)]	10^6	11.2	**4.3**

[1)] Shabib A., *et al.* Parallelization of searching and mining time series data using Dynamic Time Warping. ICACCI 2015. pp. 343–348.
[2)] Sart D., *et al.* Accelerating Dynamic Time Warping subsequence search with GPUs and FPGAs. ICDM 2010. pp. 1001–1006.
[3)] Huang S., *et al.* DTW-based subsequence similarity search on AMD heterogeneous computing platform. HPCC/EUC 2013. pp. 1054–1063.

Алгоритмы анализа временных рядов

- Поиск похожих подпоследовательностей
- **Поиск аномальных подпоследовательностей**

Поиск аномальной подпоследовательности



- Временной ряд: $T := (t_1, \dots, t_m), t_i \in \mathbb{R}$
- Подпоследовательность: $C := T_{i,n}, n \ll m$
- Непересекающаяся с C подпоследовательность: $M_C := T_{j,n}, |i - j| \geq n$
- Аномальная подпоследовательность D:

$$\forall C, M_C \in T \ \min(Euclid(D, M_D)) > \min(Euclid(C, M_C))$$

Алгоритм MDD

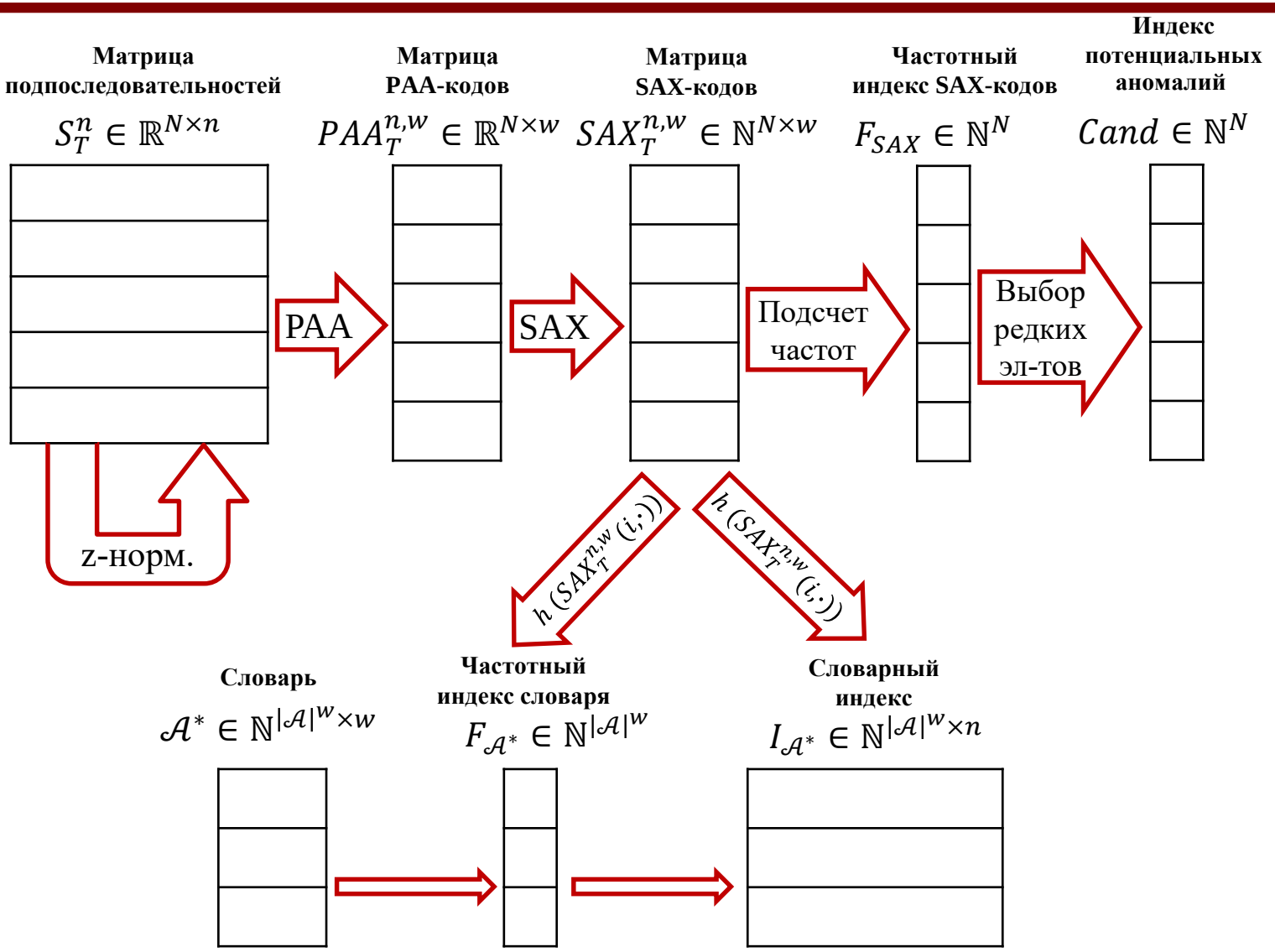
- Область применения
  - Поиск аномальной подпоследовательности временного ряда
- Платформа
  - Многоядерные ускорители
- Отличительные особенности
  - Ускорение, близкое к линейному
  - Эффективная векторизация вычислений

Алгоритм MDD

1. Построение индексных структур

2. Скоринг кандидатов

Построение индексных структур



Скоринг кандидатов

1. $anomaly \coloneqq 0$; $neighbor \coloneqq +\infty$
2. **do** $\forall C_i \in NearAnomalies$
3. **pardo** $\forall C_j \in (Neighbors \cdot Strangers)$
4. $d \coloneqq Euclid^2\ (C_i, C_j)$
5. **if** $d < anomaly$ **then**
6. **go to** 9
7. $neighbor \coloneqq \min(d, neighbor)$
8. **end pardo**
9. $anomaly \coloneqq \max(neighbor, anomaly)$
10. **if** $anomaly < neighbor$ **then**
11. $position \coloneqq i$
12. **end do**
13. **return** $\{position, anomaly\}$

Сравнение с аналогами

Алгоритм	Длина ряда	Время (сек.)	***MDD*** **(сек.)**
DDD [1)]	10^7	5 382	**1 745.3**
MR-DADD [2)]	10^6	240	**101.6**

[1)] Wu Y., *et al*. Distributed discord discovery: Spark based anomaly detection in time series. HPCC 2015. pp. 154–159.

[2)] Yankov D., *et al*. Disk aware discord discovery: Finding unusual time series in terabyte sized datasets. Knowl. Inf. Syst. 2008. vol. 17, no. 2. pp. 241–262.

Комплекс алгоритмов интеллектуального анализа данных средствами параллельной СУБД

1. Алгоритмы кластеризации
2. Алгоритмы анализа временных рядов
3. **Алгоритмы поиска шаблонов**

Задача поиска шаблонов

T_1: {сыр, хлеб} T_2: {сыр, хлеб, чай} T_3: {кофе, сыр, чай} T_4: {кофе, чай} T_n: {кофе, сахар, сыр, чай}

В супермаркете *найти* **все наборы товаров**, которые **часто** покупают **совместно**

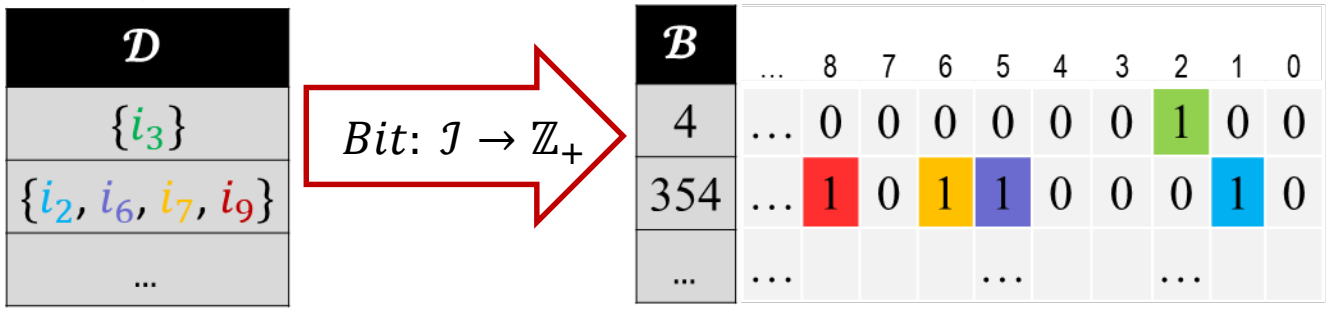
Множество объектов	$\mathcal{I} \coloneqq \{i_1, i_2, \dots, i_m\}$
k-набор объектов	$I \subseteq \mathcal{I}, \lvert I \rvert = k$
Множество транзакций (корзин)	$\mathcal{D} \coloneqq \{T_1, \dots, T_n\}, \forall i\ T_i = I, I \subseteq \mathcal{I}$
Поддержка набора	$supp(I) \coloneqq \frac{\lvert\{T_i \in \mathcal{D} \mid I \subseteq T_i\}\rvert}{n}$
Порог поддержки	$minsup\ (0 < minsup \leq 1)$
Множество частых k-наборов	$\mathcal{L}_k \coloneqq \{I \mid supp(I) \geq minsup\ \wedge \lvert I \rvert = k\}$
Множество всех частых наборов	$\mathcal{L} \coloneqq \cup_{k=1}^{k_{max}} \mathcal{L}_k$

Алгоритм PDIC

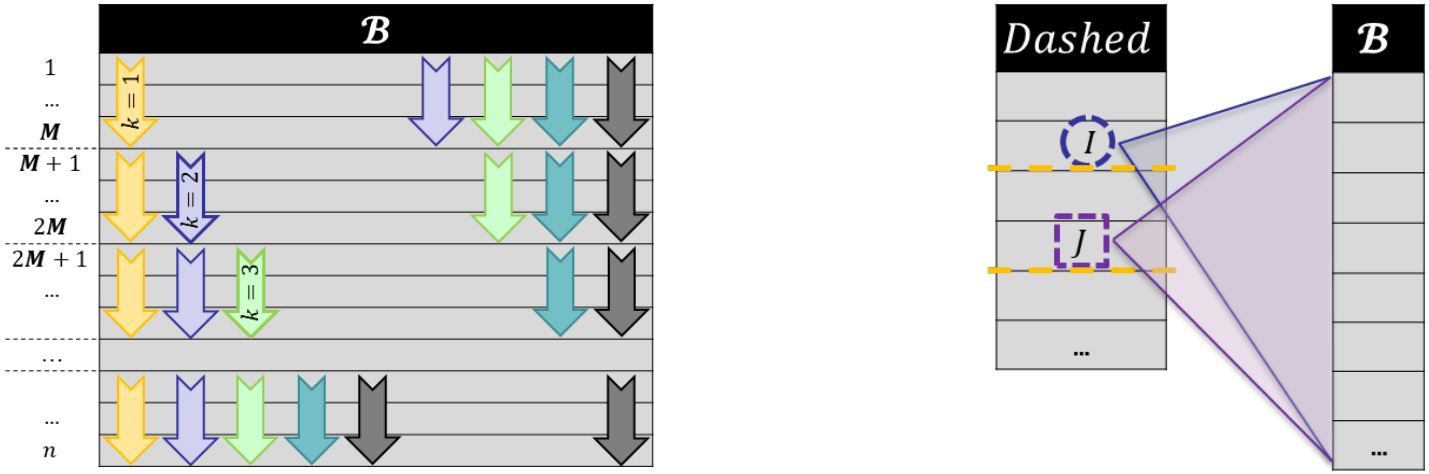
- Область применения
  - Поиск частых наборов в сверхбольших данных
- Платформа
  - Многоядерные ускорители
- Отличительные особенности
  - Использование битовых масок
  - Эффективная векторизация
  - Линейное ускорение

Структура алгоритма PDIC

1. Битовое представление множества транзакций (корзин)



2. Подсчет поддержки наборов



Битовое представление множества транзакций (корзин)

Множество транзакций

$\mathcal{D}$
$\{i_3\}$
$\{i_2, i_6, i_7, i_9\}$
…

Функция битовой маски

$Bit: \mathcal{I} \to \mathbb{Z}_+$

Битовая карта множества транзакций

$\mathcal{B}$	…	8	7	6	5	4	3	2	1	0
4	…	0	0	0	0	0	0	**1**	0	0
354	…	**1**	0	**1**	**1**	0	0	0	**1**	0
…	…				…			…		

Подсчет поддержки набора

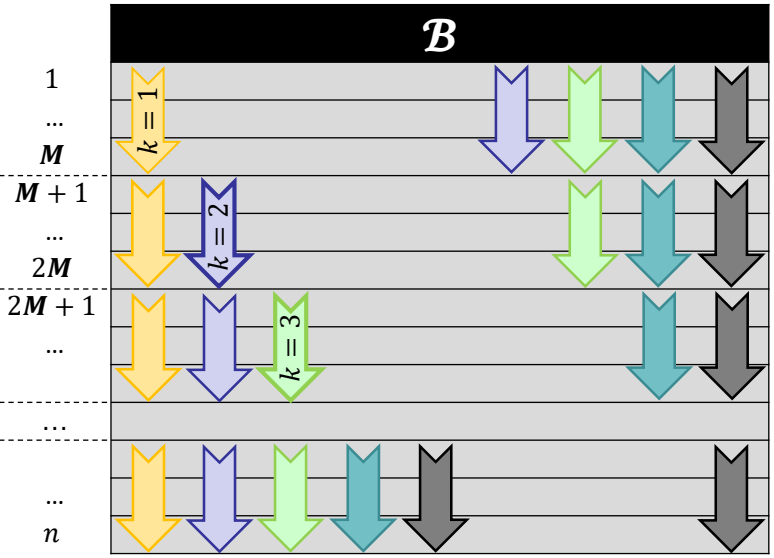
$$I \subseteq T \Leftrightarrow Bit(I) \equiv Bit(I) \wedge Bit(T)$$

Генерация нового k-набора для подсчета поддержки

$$C_k := Bit(I) \vee Bit(J)$$

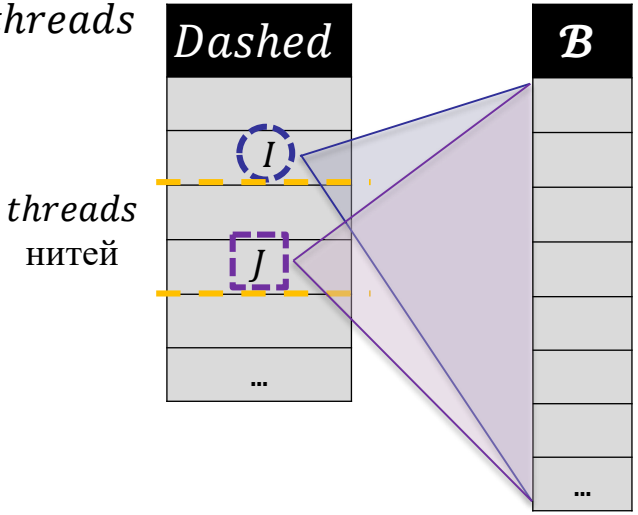
$$\forall I, J \in \{I, J \subseteq \mathcal{I} \mid I \neq J, |I| = |J| = k, |I \cap J| = k - 1\}$$

Подсчет поддержки

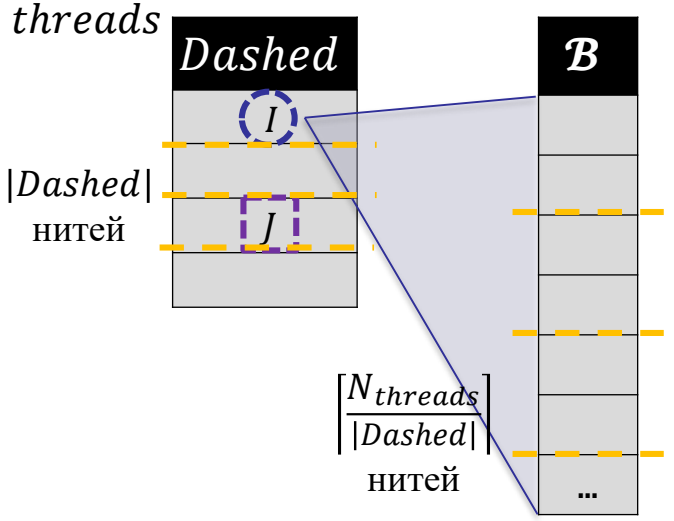


- Функция-счетчик транзакций: $Count: \mathcal{I} \times \mathcal{B} \to \mathbb{N} \cup \{0\}$
- Множества наборов
  - ***Неподтвержденные редкие*** $DashedCircle := \{I \mid Count(I) < n \ \wedge supp(I) < minsup\}$
  - ***Неподтвержденные частые*** $DashedBox := \{I \mid Count(I) < n \ \wedge supp(I) \geq minsup\}$
  - ***Подтвержденные частые*** $SolidBox := \{I \mid Count(I) = n \ \wedge supp(I) \geq minsup\}$
  - ***Подтвержденные редкие*** $SolidCircle := \{I \mid Count(I) = n \wedge supp(I) < minsup\}$
  - ***Неподтвержденные:*** $Dashed := DashedCircle \cup DashedBox$
  - ***Подтвержденные:*** $Solid := SolidCircle \cup SolidBox$

А. $|Dashed| \geq threads$



Б. $|Dashed| < threads$



Сравнение с аналогами [1)]

Алгоритм	Время (сек.)	
	Набор RANDOM [2)] $\lvert\mathcal{D}\rvert = 2 \cdot 10^7$	Набор TORNADO [3)] $\lvert\mathcal{D}\rvert = 2 \cdot 10^7$
Apriori	166.6	7.6
FP-Growth	66.7	28.6
Eclat	73.7	31.6
PDIC	**44.1**	**4.0**

[1)] Borgelt C. Frequent Item Set Mining. WIREs: Data Mining and Knowledge Discovery. 2012. vol. 2, no. 6, pp. 437–456.
[2)] IBM Quest Synthetic Data Generator. URL: https://ibmquestdatagen.sourceforge.io/.
[3)] Zymbler M. Parallel algorithm for frequent itemset mining on Intel manycore systems. CIT. 2019. vol. 26, no. 4. pp. 209–221.

Основные результаты, выносимые на защиту

1. Разработан программный комплекс для интеллектуального анализа данных средствами СУБД на кластерных вычислительных системах с многоядерными ускорителями, включающий следующие параллельные алгоритмы:
   - алгоритм кластеризации графа для параллельной СУБД на основе фрагментного параллелизма;
   - алгоритм нечеткой кластеризации данных для параллельной СУБД на основе фрагментного параллелизма;
   - параллельный алгоритм поиска похожих подпоследовательностей временного ряда для кластеров с многоядерными ускорителями;
   - параллельный алгоритм поиска аномальных подпоследовательностей временного ряда для многоядерных ускорителей;
   - параллельный алгоритм кластеризации данных для многоядерных ускорителей на основе техники медоидов;
   - параллельный алгоритм поиска частых наборов для многоядерных ускорителей на основе битовых масок
2. Предложен метод внедрения параллелизма в свободную СУБД и разработан прототип параллельной СУБД на его основе

Спасибо за внимание!

Публикации по теме диссертации

- **12 статей** в журналах Перечня ВАК
- **9 статей** в изданиях Scopus и WoS
- **3 свидетельства** о регистрации программ

Апробации

17 докладов на международных и всероссийских научных конференциях (2010–2019 гг.)

ADBIS 2013 и 2015, DEXA 2013, SISAP 2015,
AICT 2015, DAMDID 2017–2019,
ПаВТ 2010–2019, Суперкомпьютерные дни в России 2015

Репозиторий исходных текстов

Тексты программ, реализующих предложенные методы и алгоритмы, размещены по адресу

https://github.com/mzym/thesis/