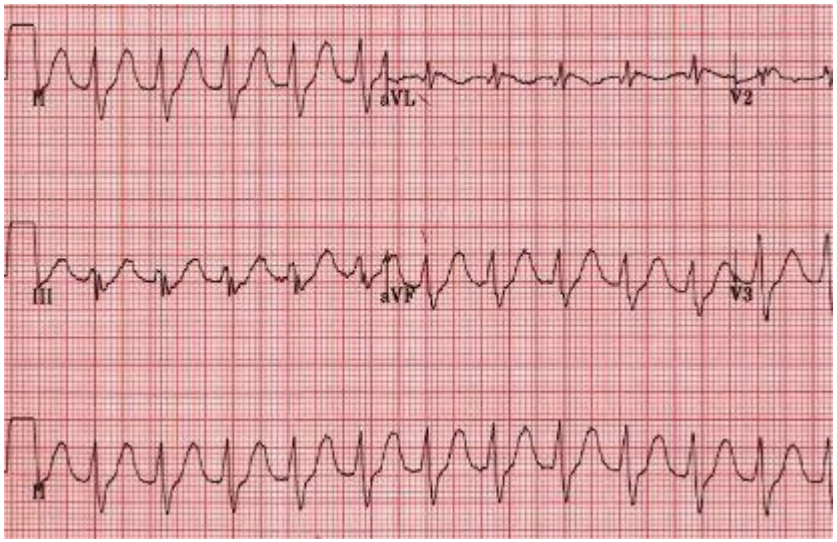
Поиск похожих подпоследовательностей временных рядов на сопроцессорах Intel Xeon Phi

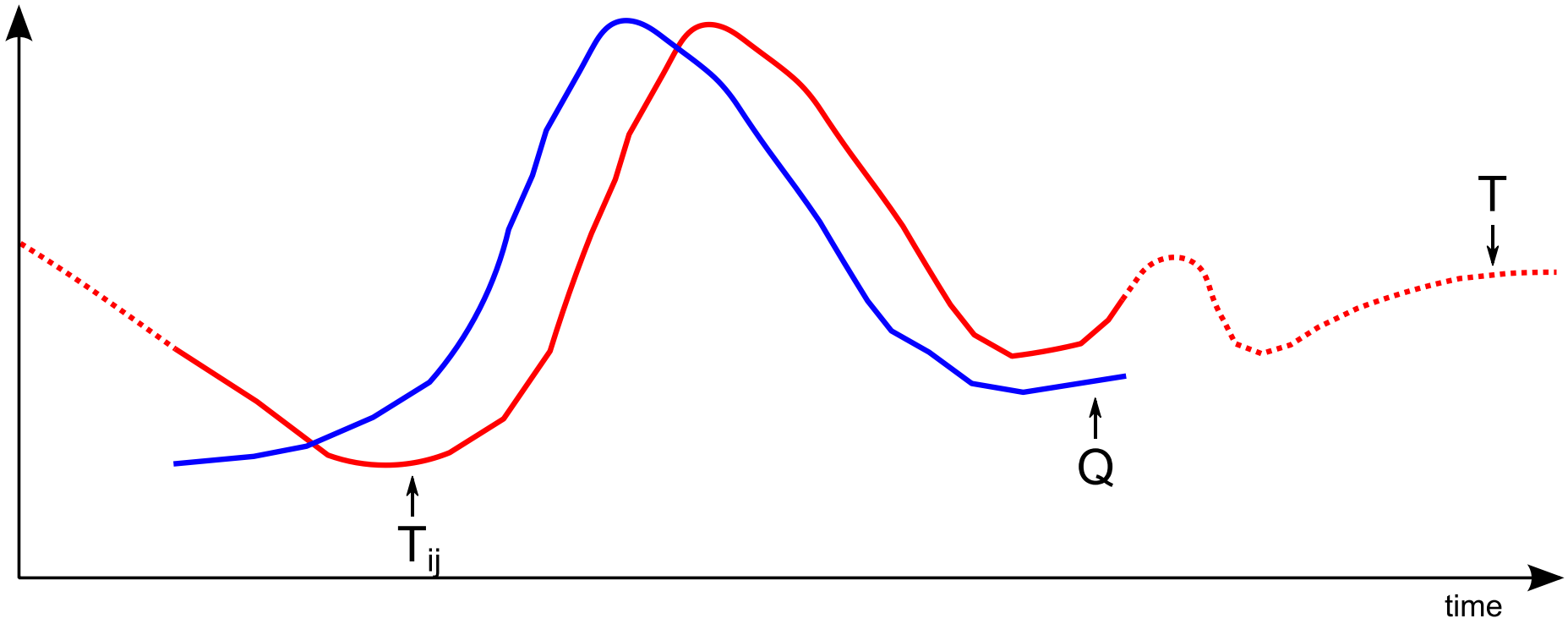
М.Л. Цымблер, А.В. Мовчан
Отдел интеллектуального анализа данных
Лаборатории суперкомпьютерного моделирования НИУ ЮУрГУ
Челябинск
кандидат физ.-мат. наук, доцент
mzym@susu.ru

Конференция «Большие Данные в национальной экономике», Москва, 21 октября 2014

Временной ряд



Поиск подпоследовательностей



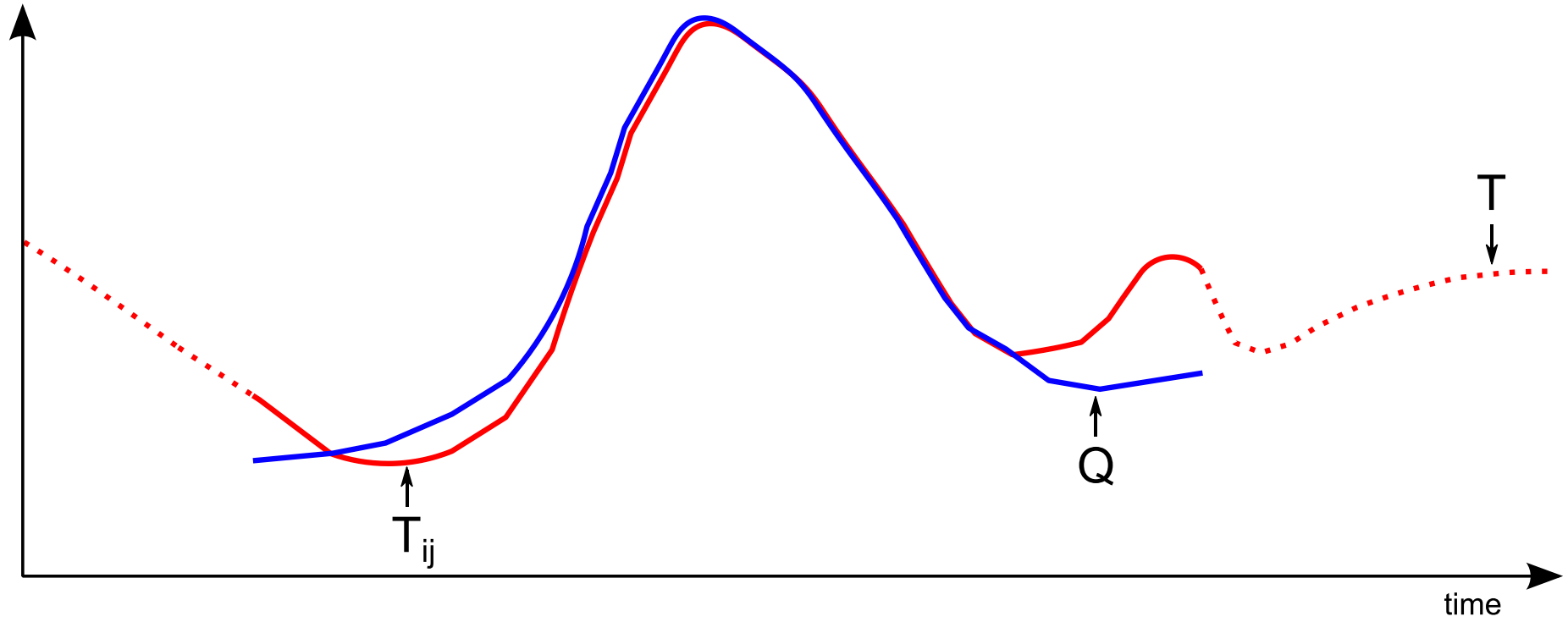
T – временной ряд, по которому осуществляется поиск

T_{ij} – подпоследовательность

Q – запрос

D – функция расстояния

Поиск подпоследовательностей



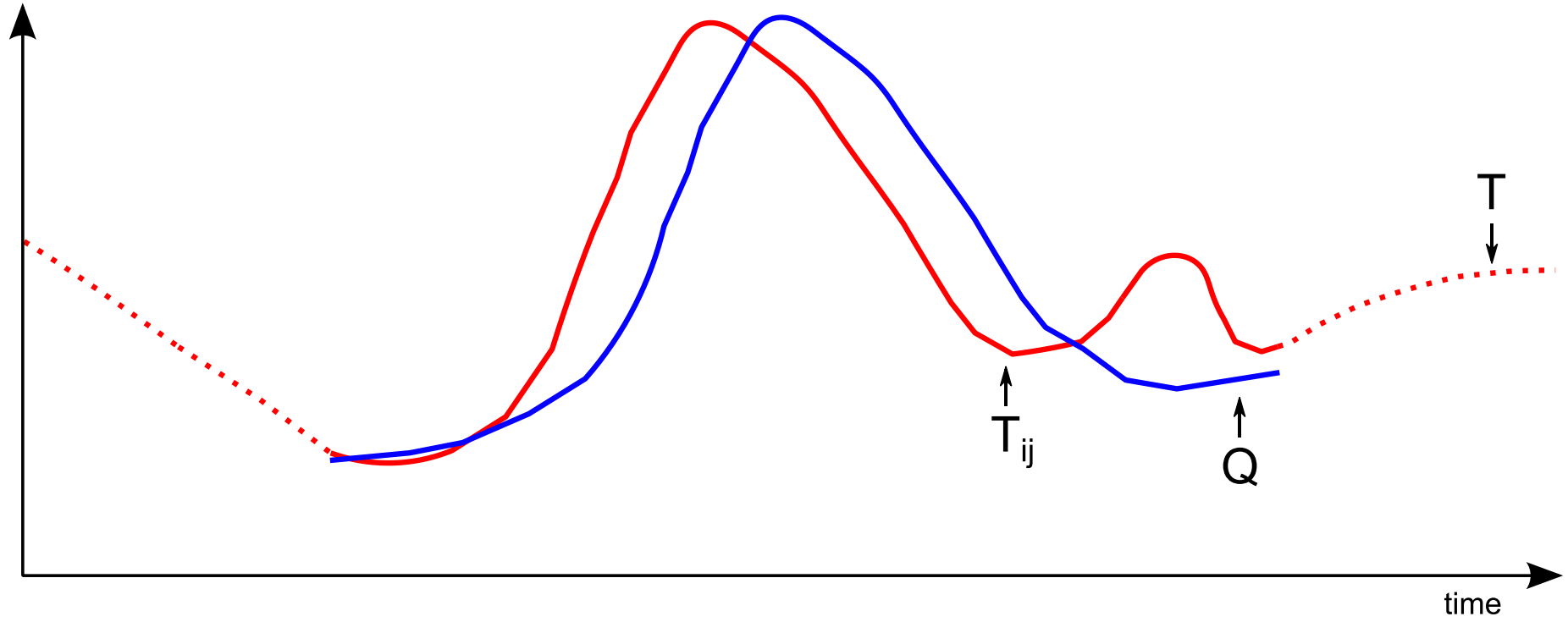
T – временной ряд, по которому осуществляется поиск

T_{ij} – подпоследовательность

Q – запрос

D – функция расстояния

Поиск подпоследовательностей



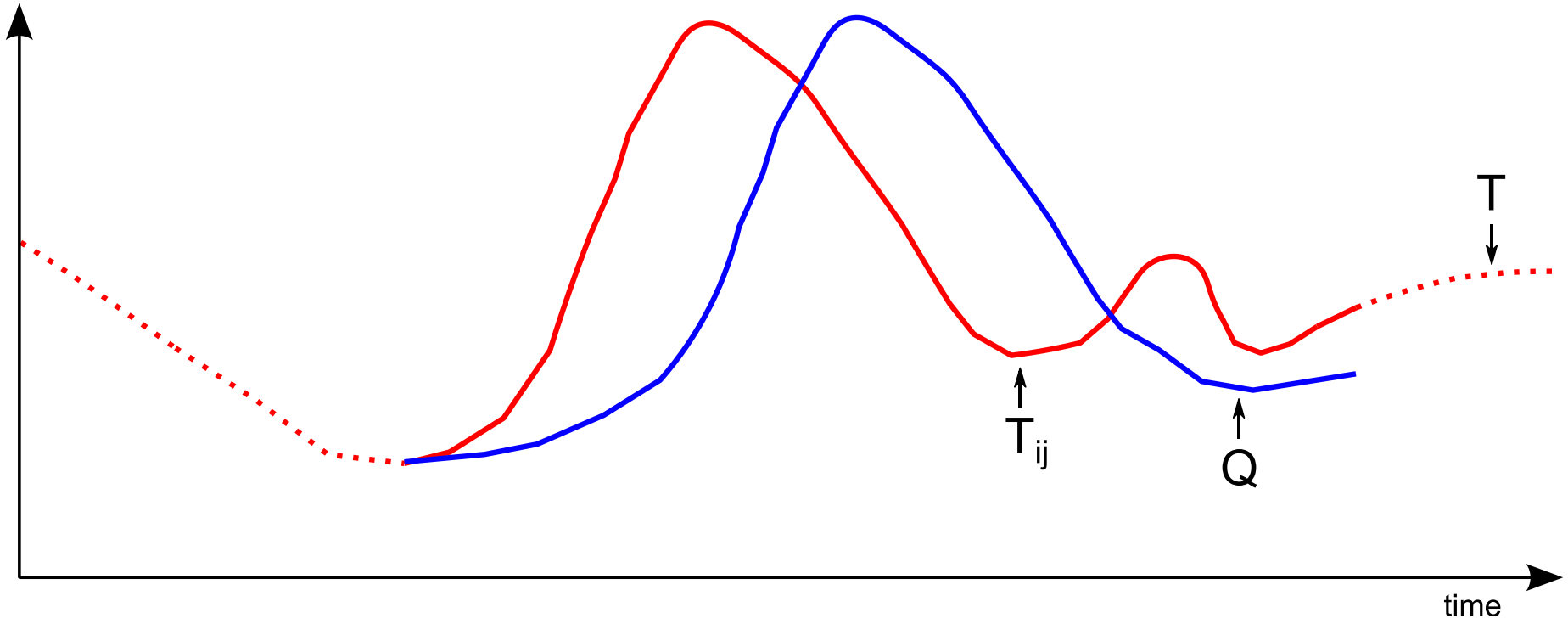
T – временной ряд, по которому осуществляется поиск

T_{ij} – подпоследовательность

Q – запрос

D – функция расстояния

Поиск подпоследовательностей



T – временной ряд, по которому осуществляется поиск

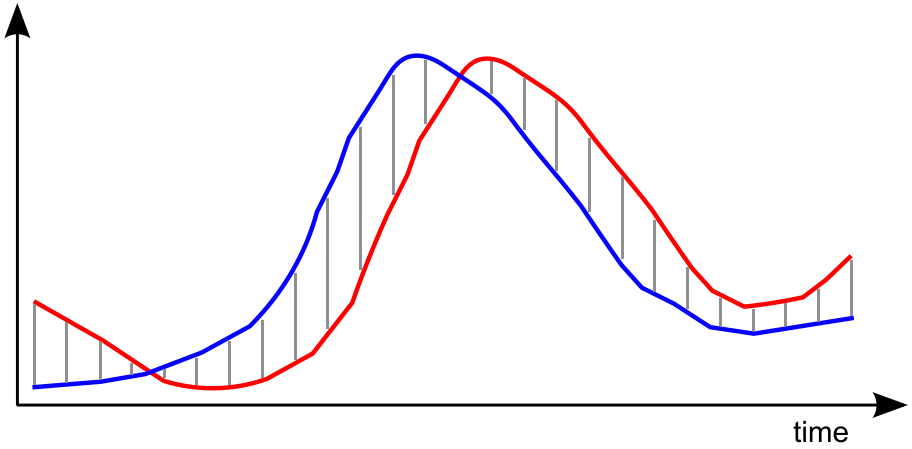
T_{ij} – подпоследовательность

Q – запрос

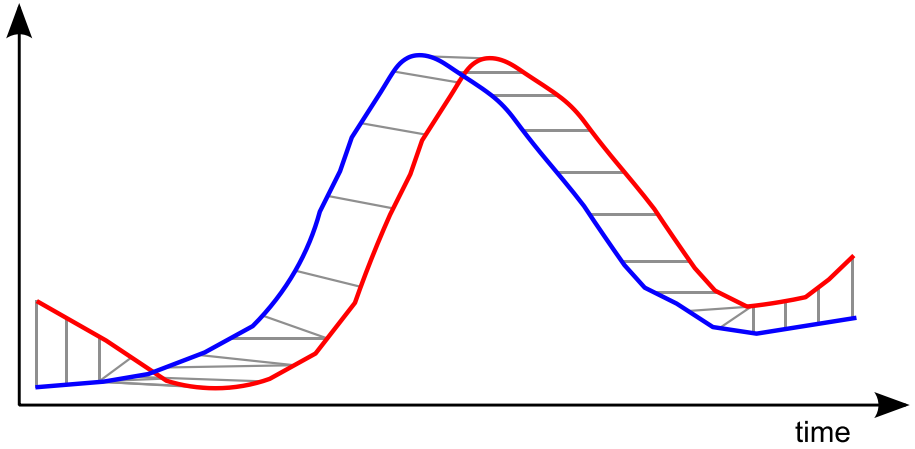
D – функция расстояния

Схожесть временных рядов

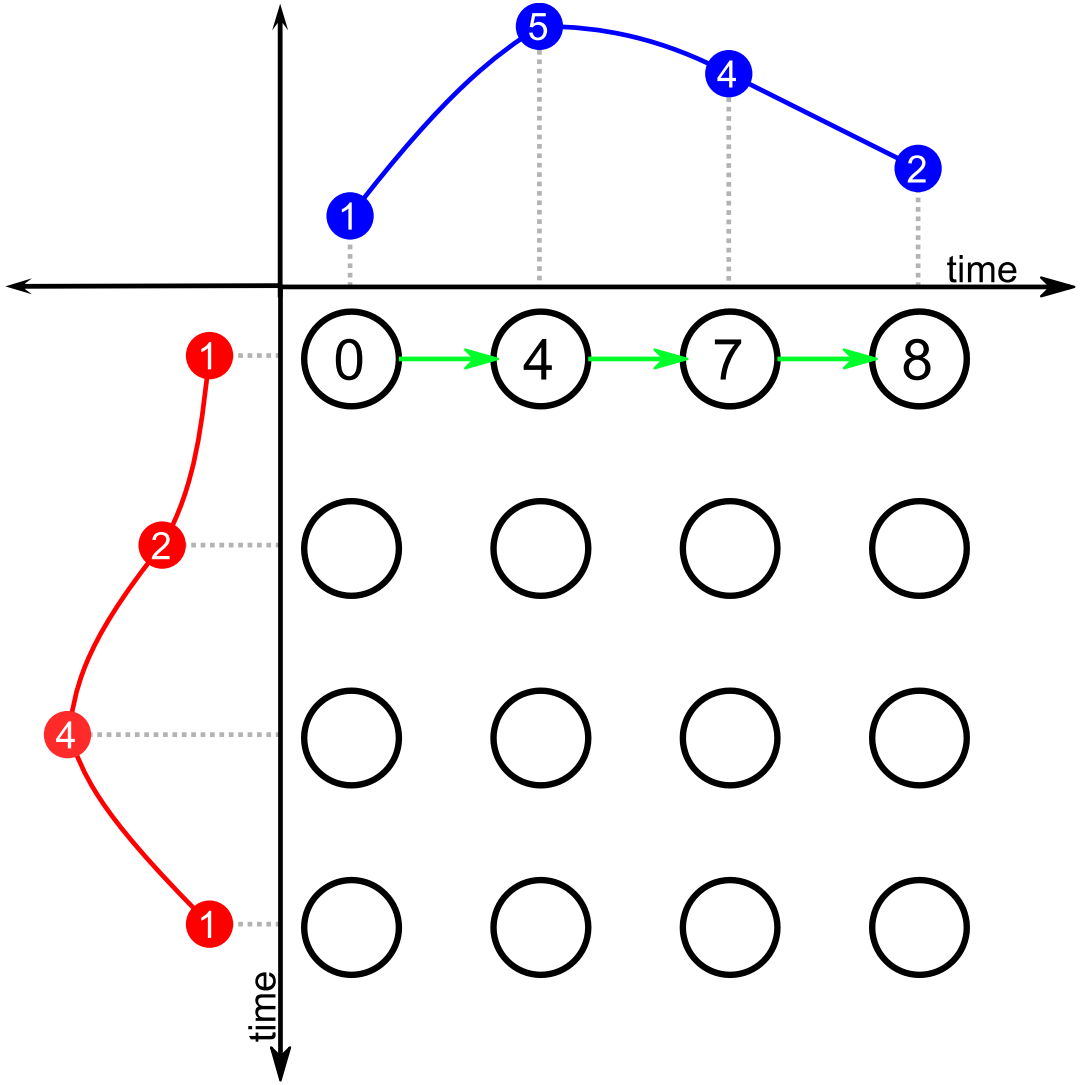
Евклидова метрика

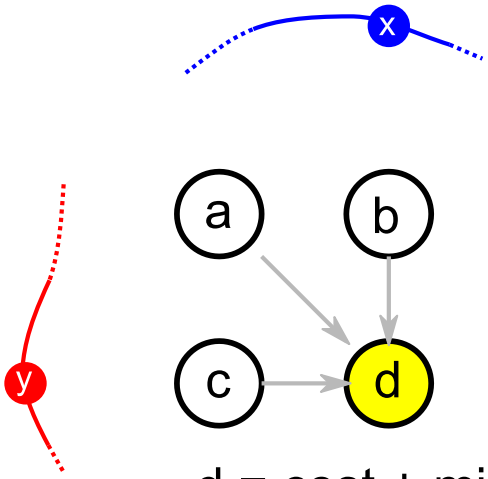


Динамическая трансформация шкалы времени (DTW, Dynamic Time Warping)



Dynamic Time Warping

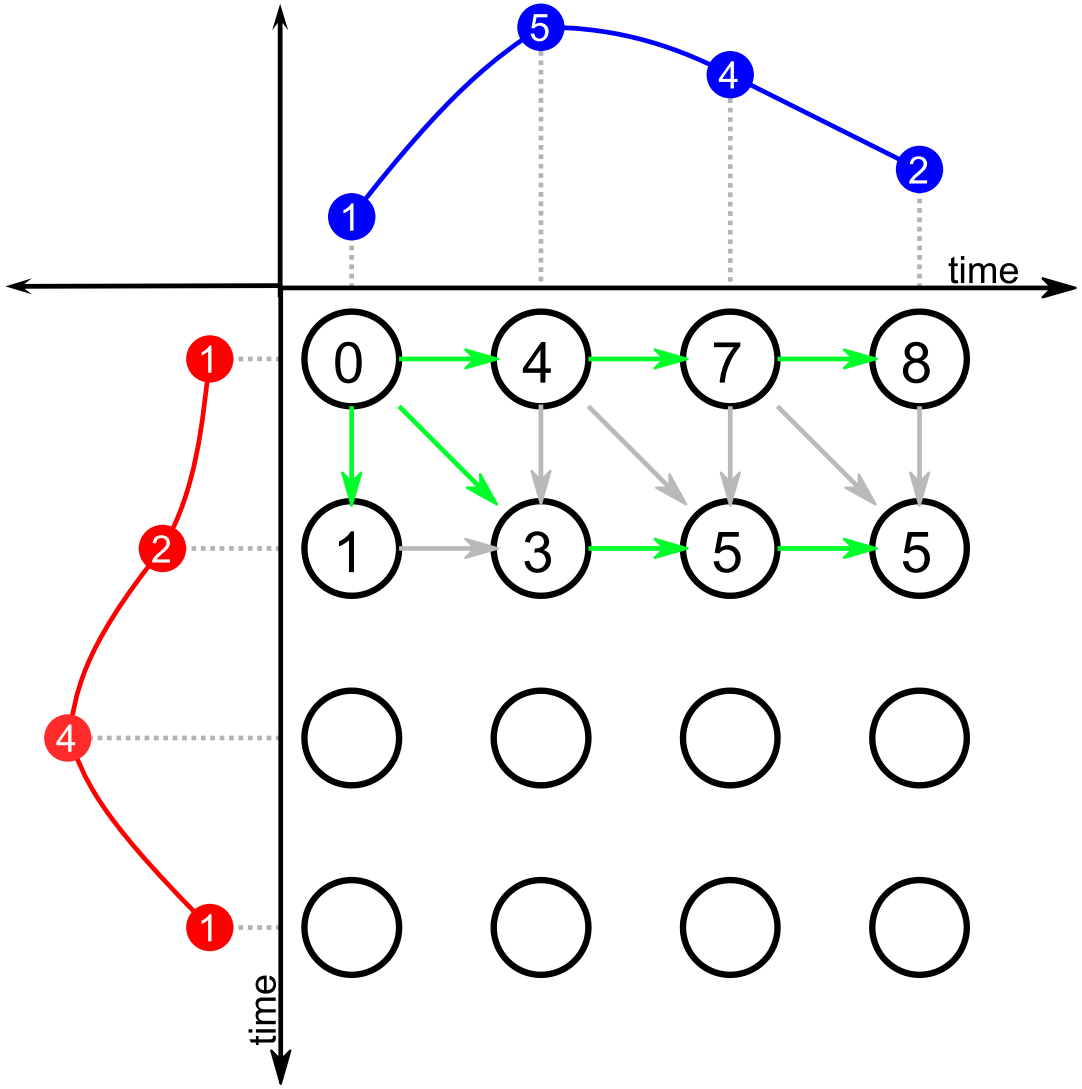


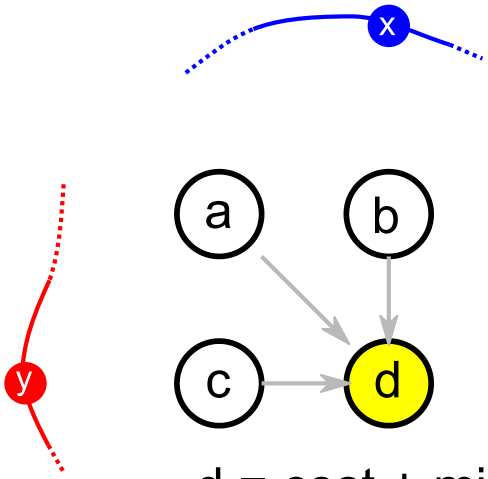


d = cost + min(a, b, c)

cost = |x - y|

Dynamic Time Warping

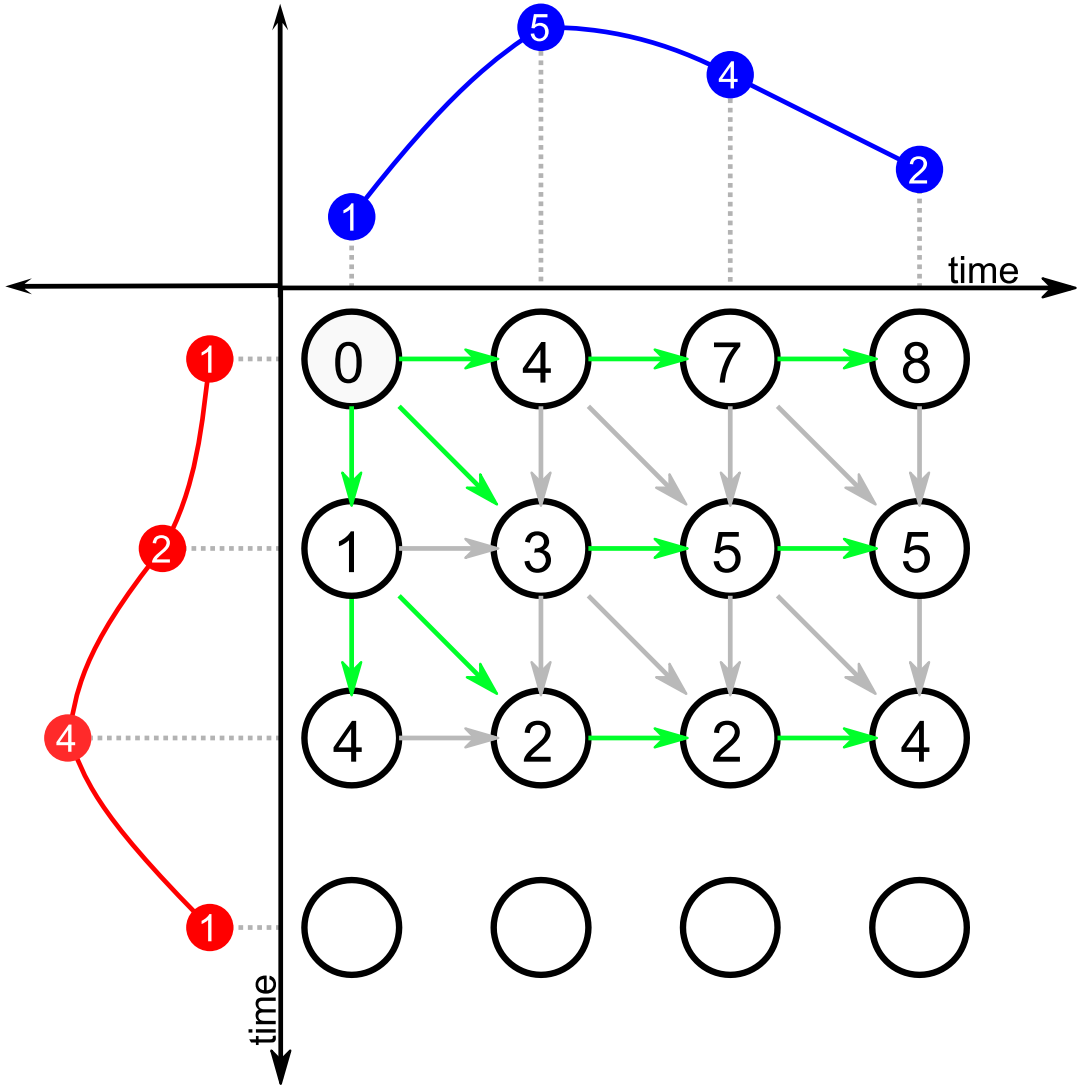


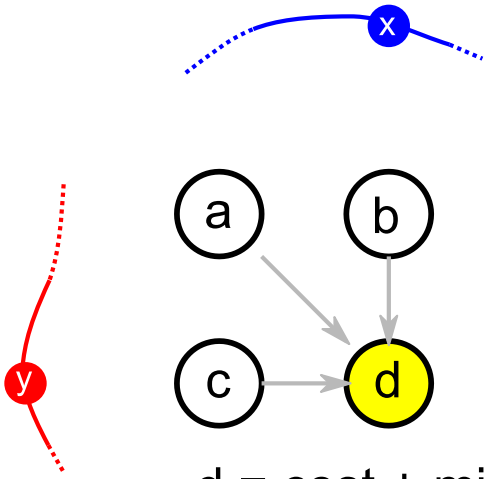


d = cost + min(a, b, c)

cost = |x - y|

Dynamic Time Warping

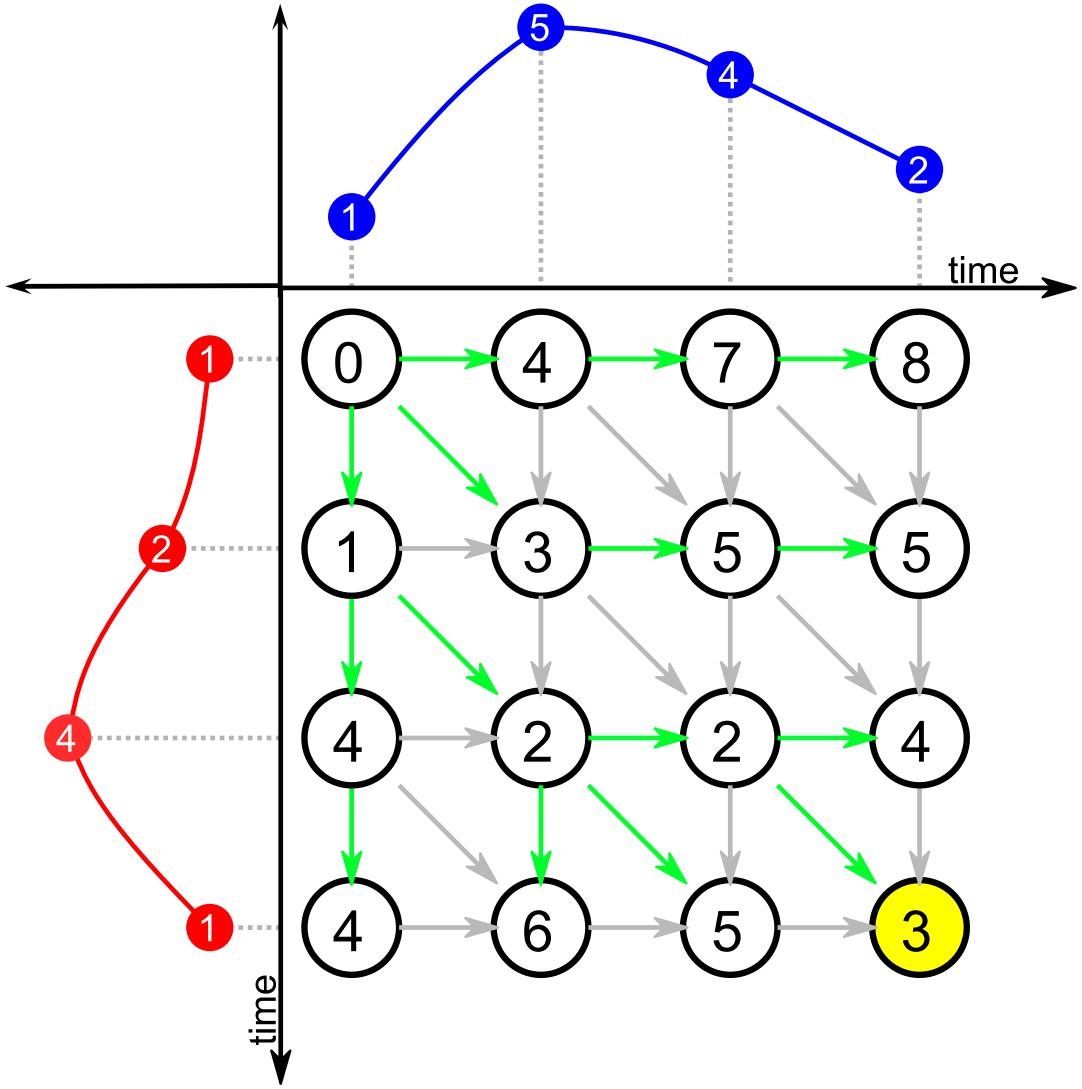


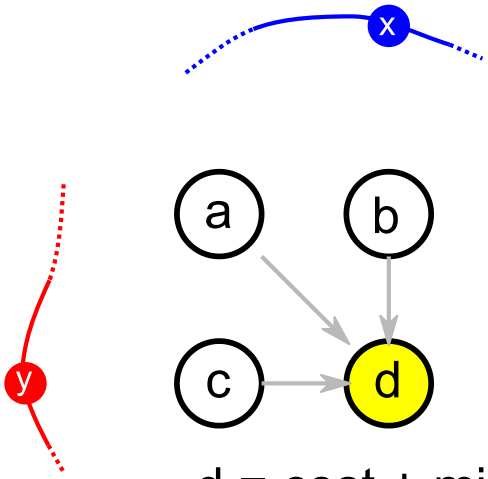


d = cost + min(a, b, c)

cost = |x - y|

Dynamic Time Warping



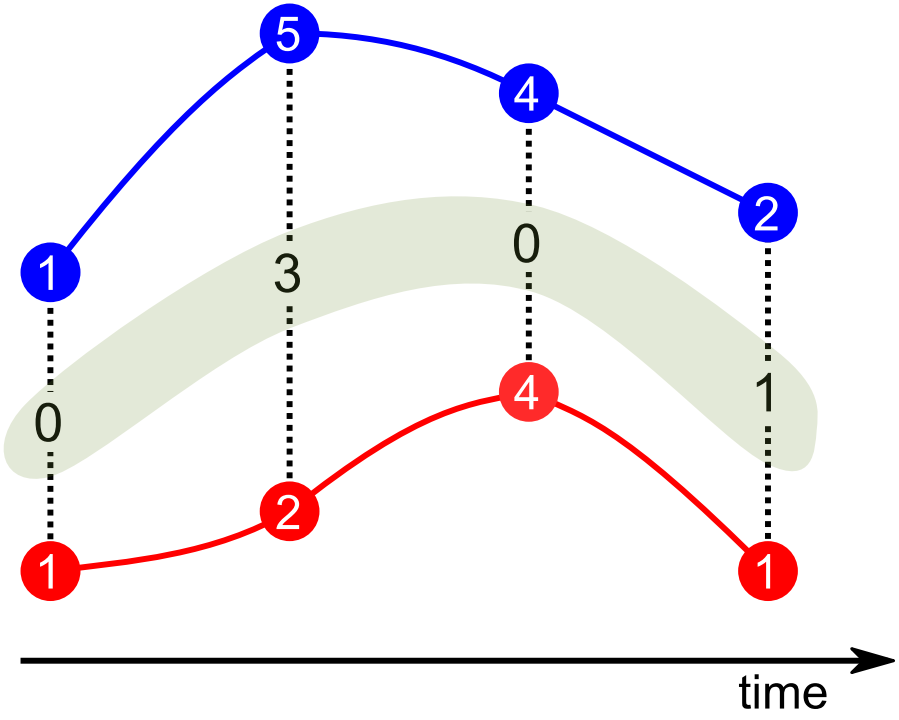


d = cost + min(a, b, c)

cost = |x - y|

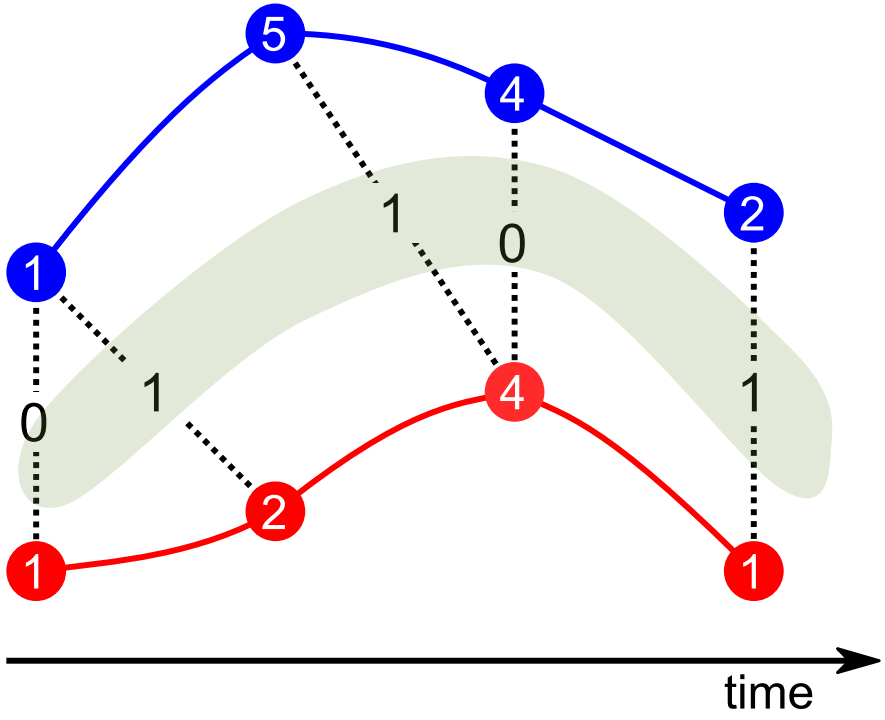
Dynamic Time Warping

Евклидова метрика



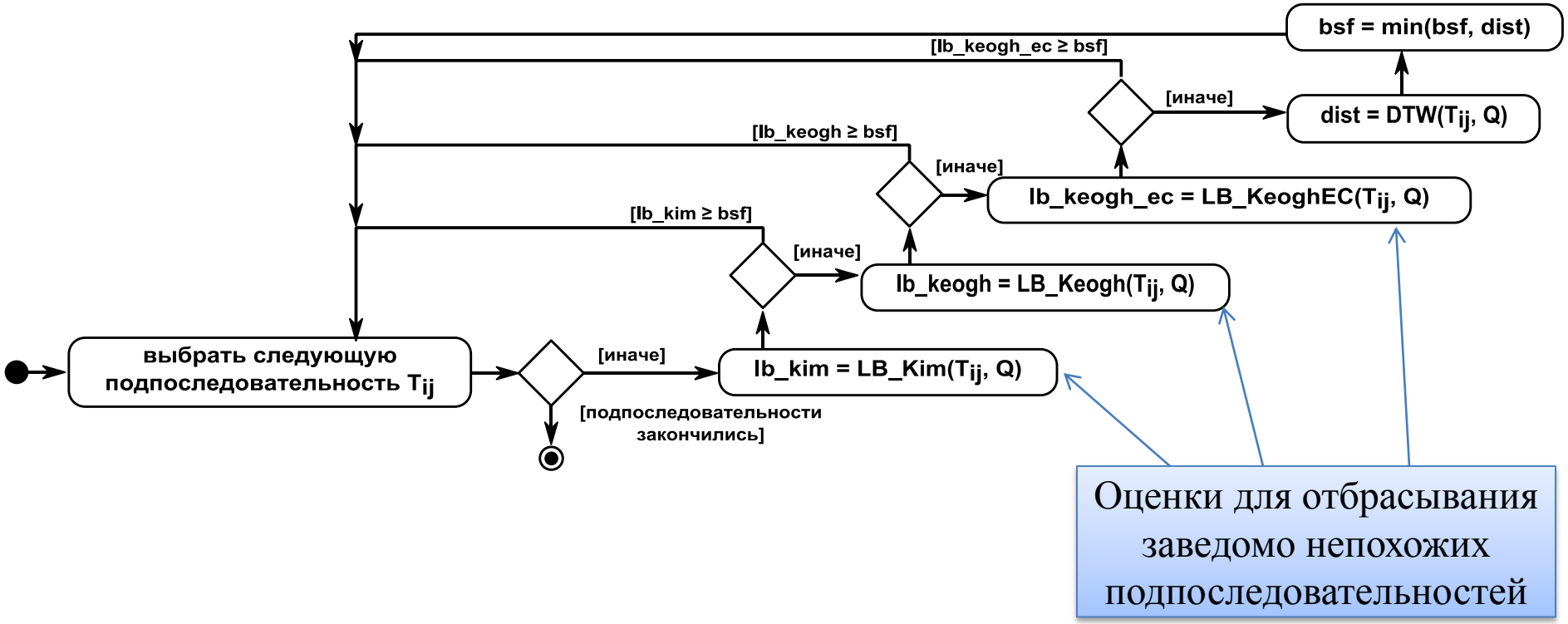
расстояние = 4

Динамическая трансформация шкалы времени



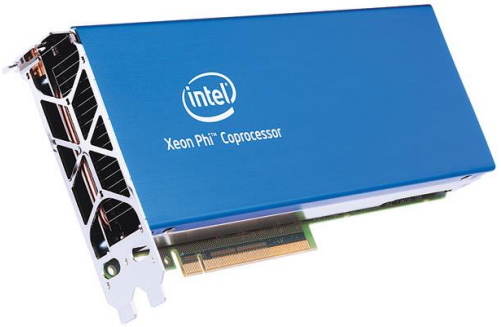
расстояние = 3

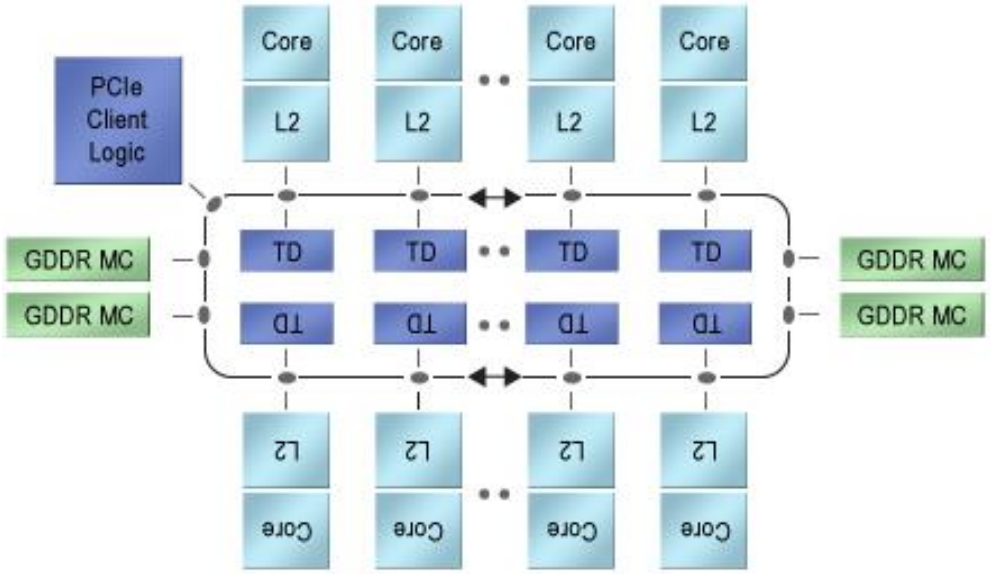
Последовательный алгоритм UCR-DTW



Rakthanmanon T., et al. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping // The 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Beijing, China, 12-16 August, 2012. ACM, 2012. P. 262–270.

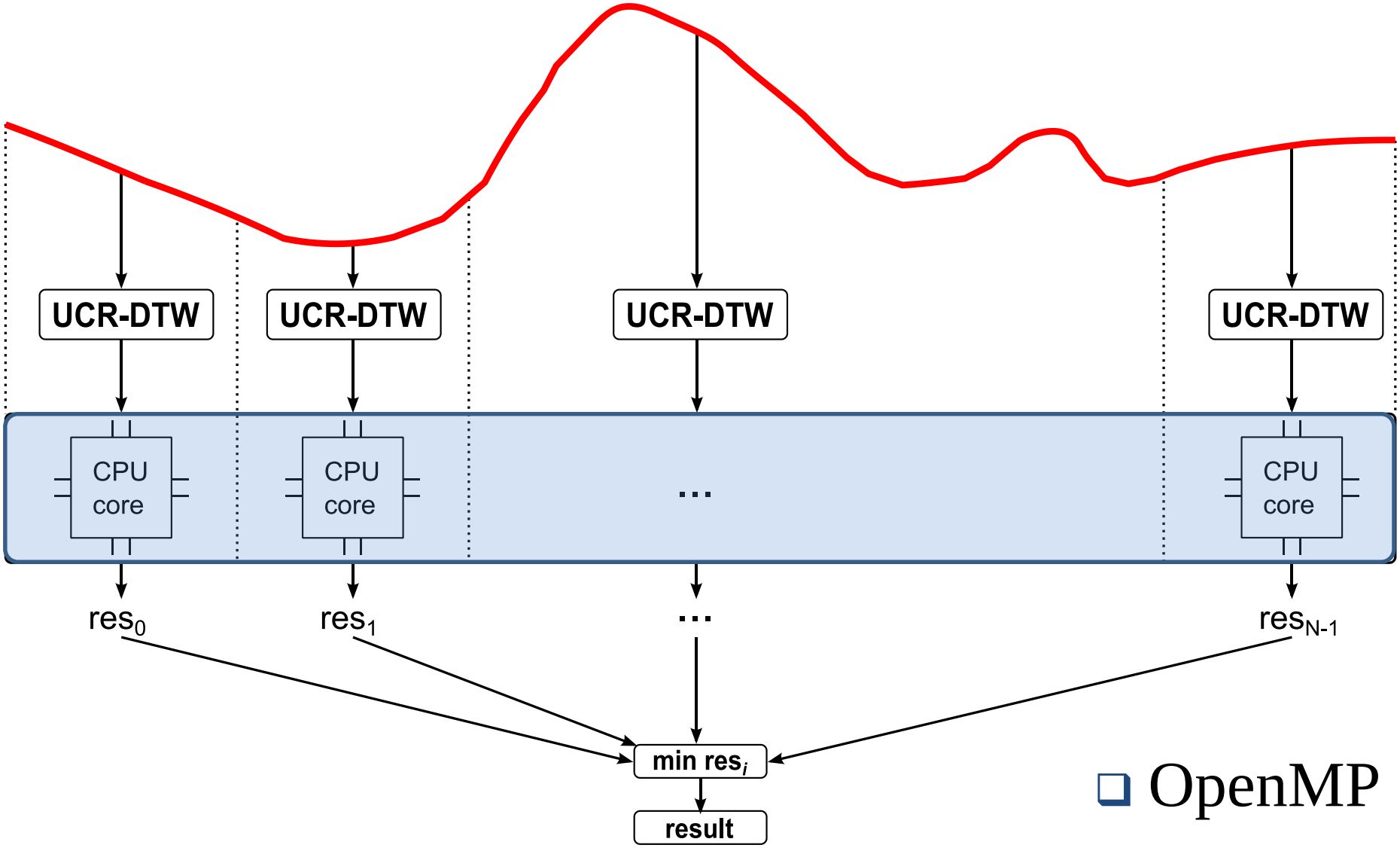
Сопроцессор Intel Xeon Phi





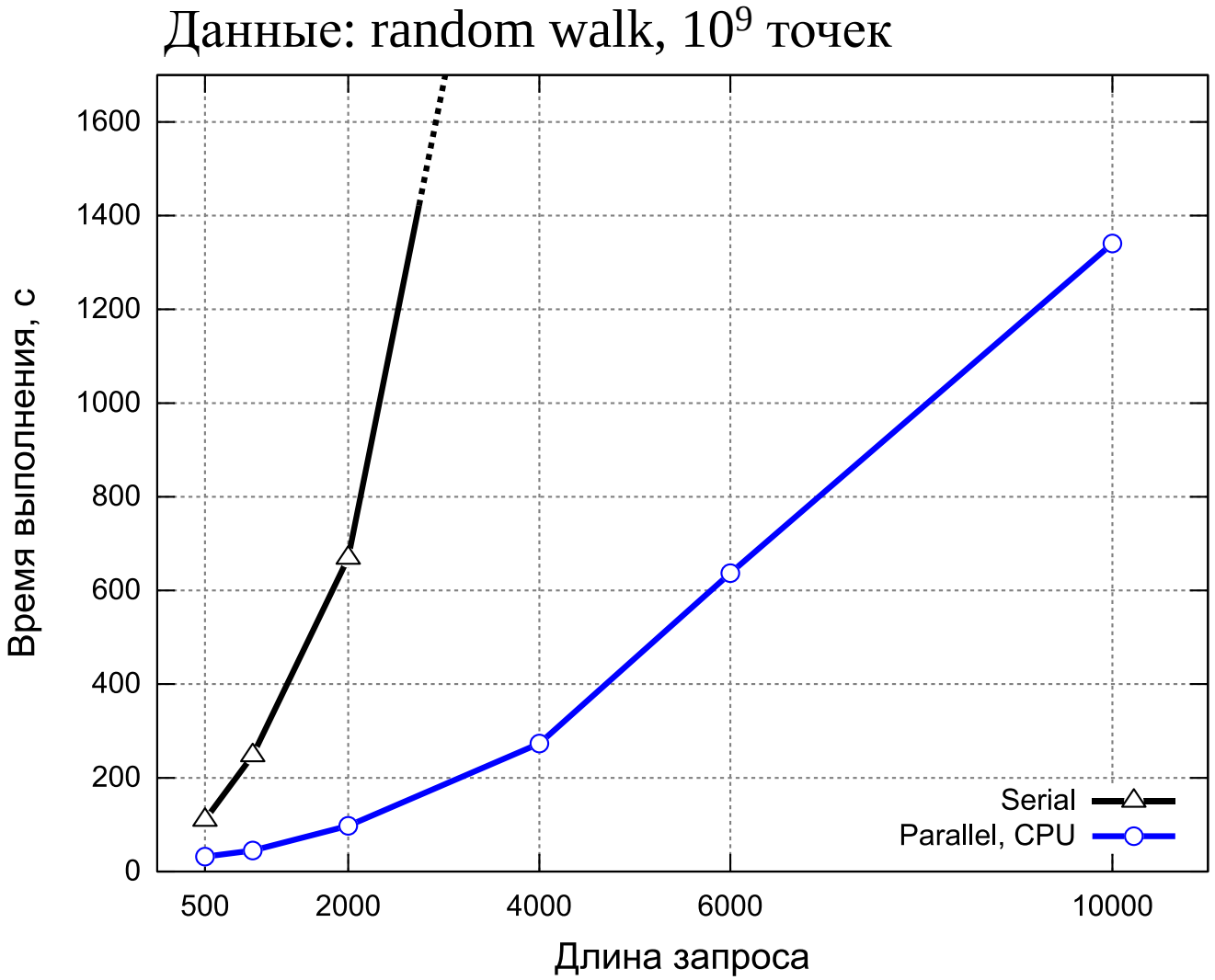
- Native mode
  - независимое исполнение на сопроцессоре
- Offload mode
  - исполнение на процессоре, выгрузка интенсивных вычислений на сопроцессор
- Symmetric mode
  - исполнение как приложения MPI

Параллельный алгоритм для CPU

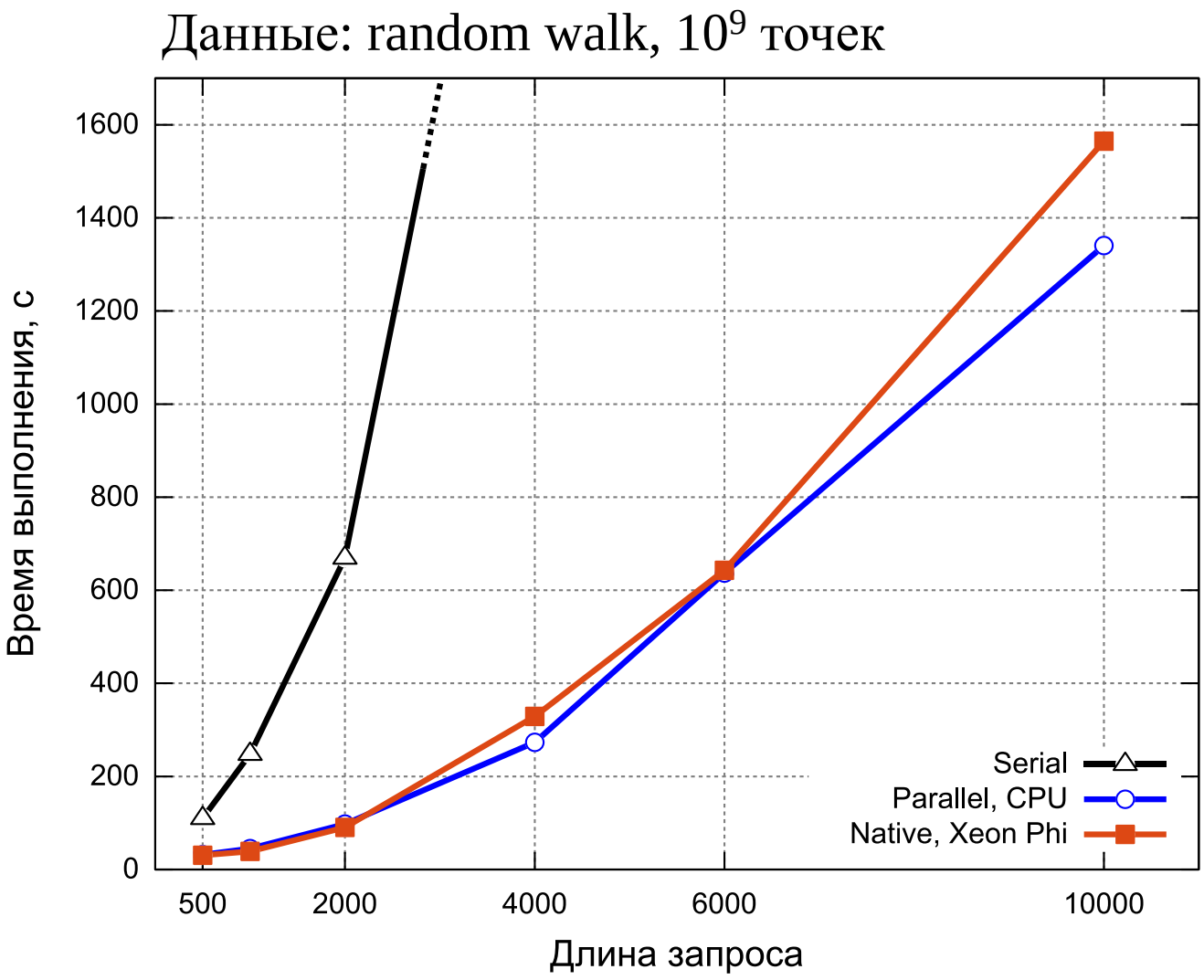


- OpenMP

Результаты экспериментов, параллельный алгоритм для CPU



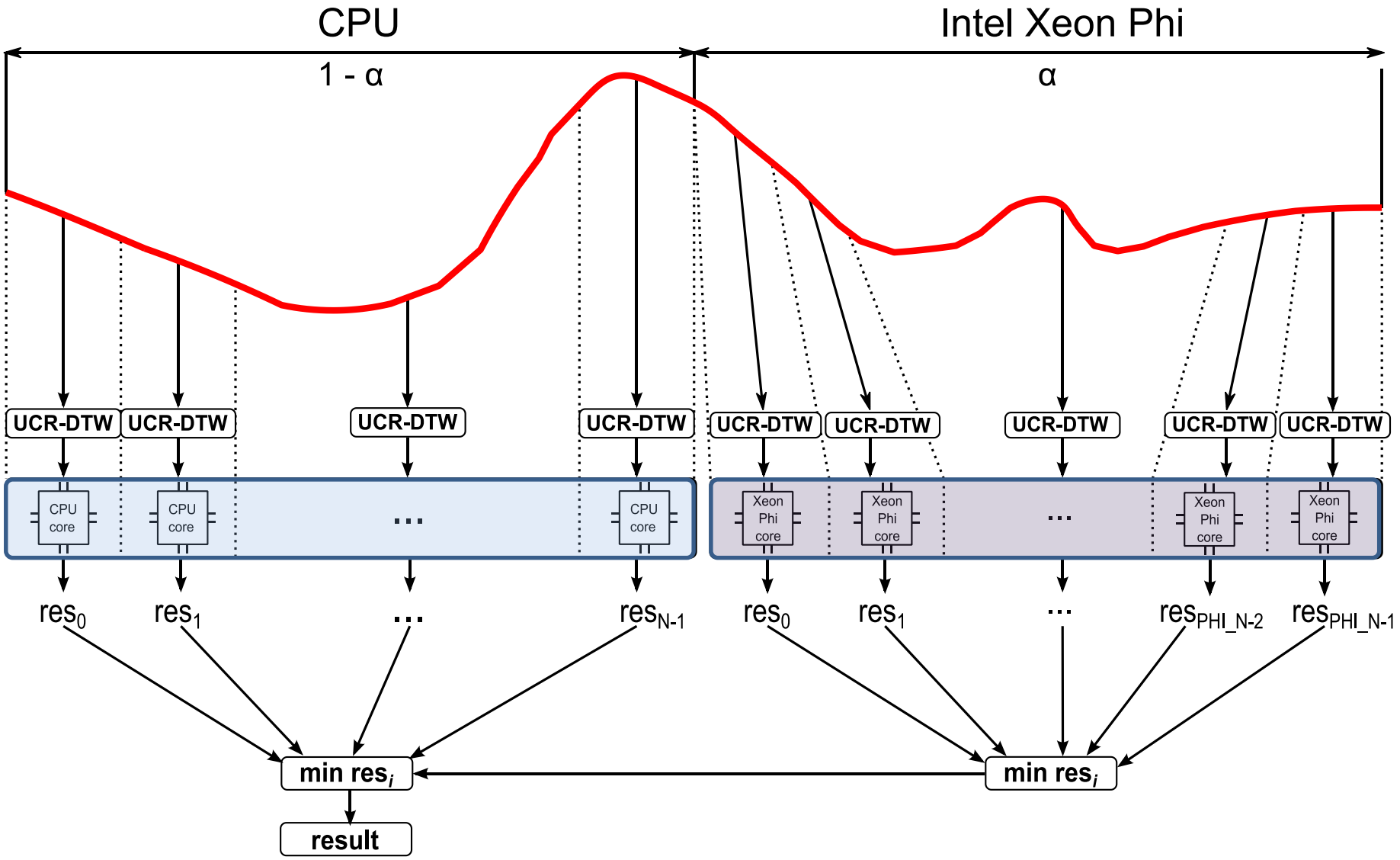
Результаты экспериментов, native mode



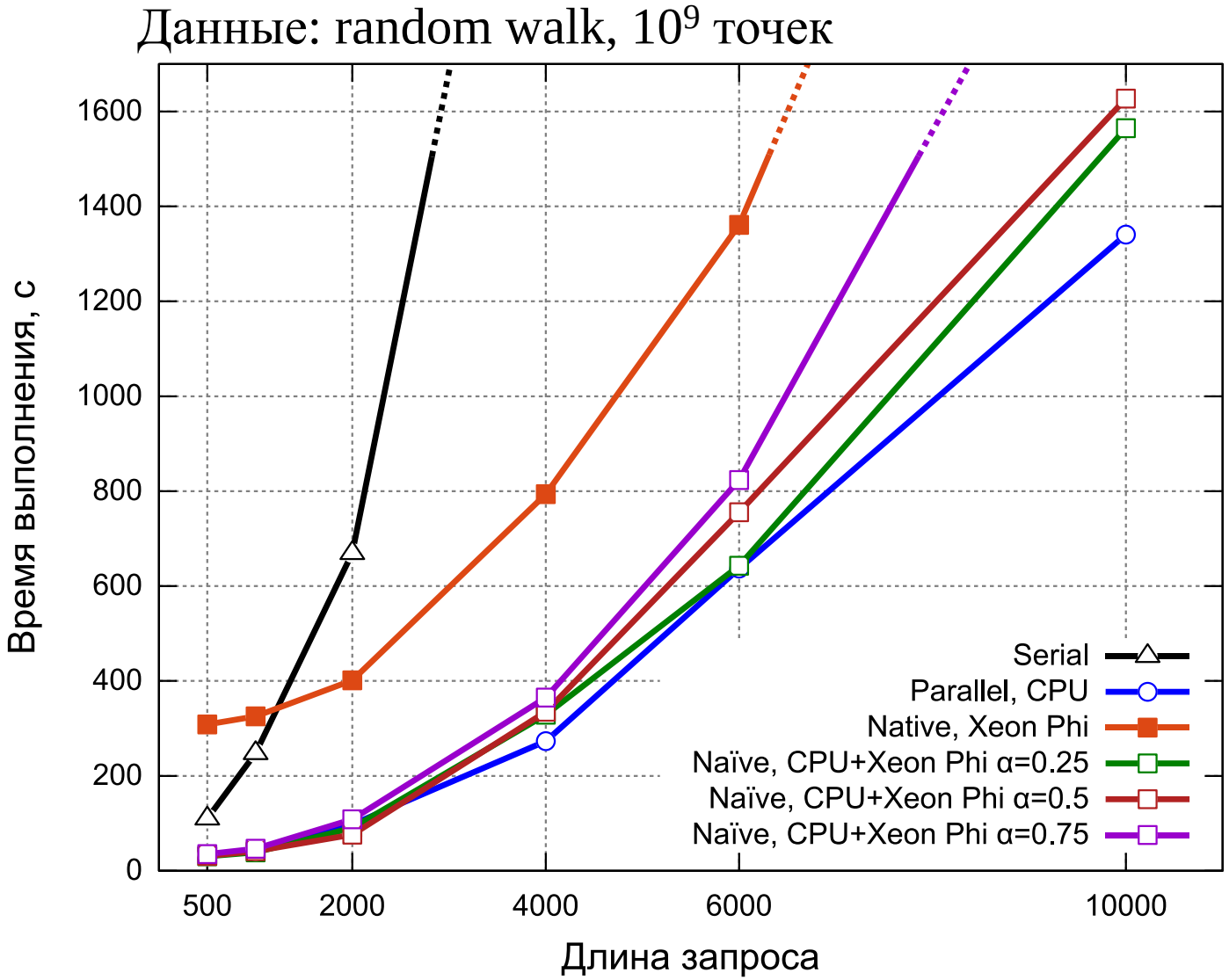
LB_Kim	O(1)
LB_Keogh	O(n)
LB_KeoghEC	O(n)
DTW	O(n^2)

Время загрузки
в память Phi: ≈300 с

Наивный алгоритм для Xeon Phi



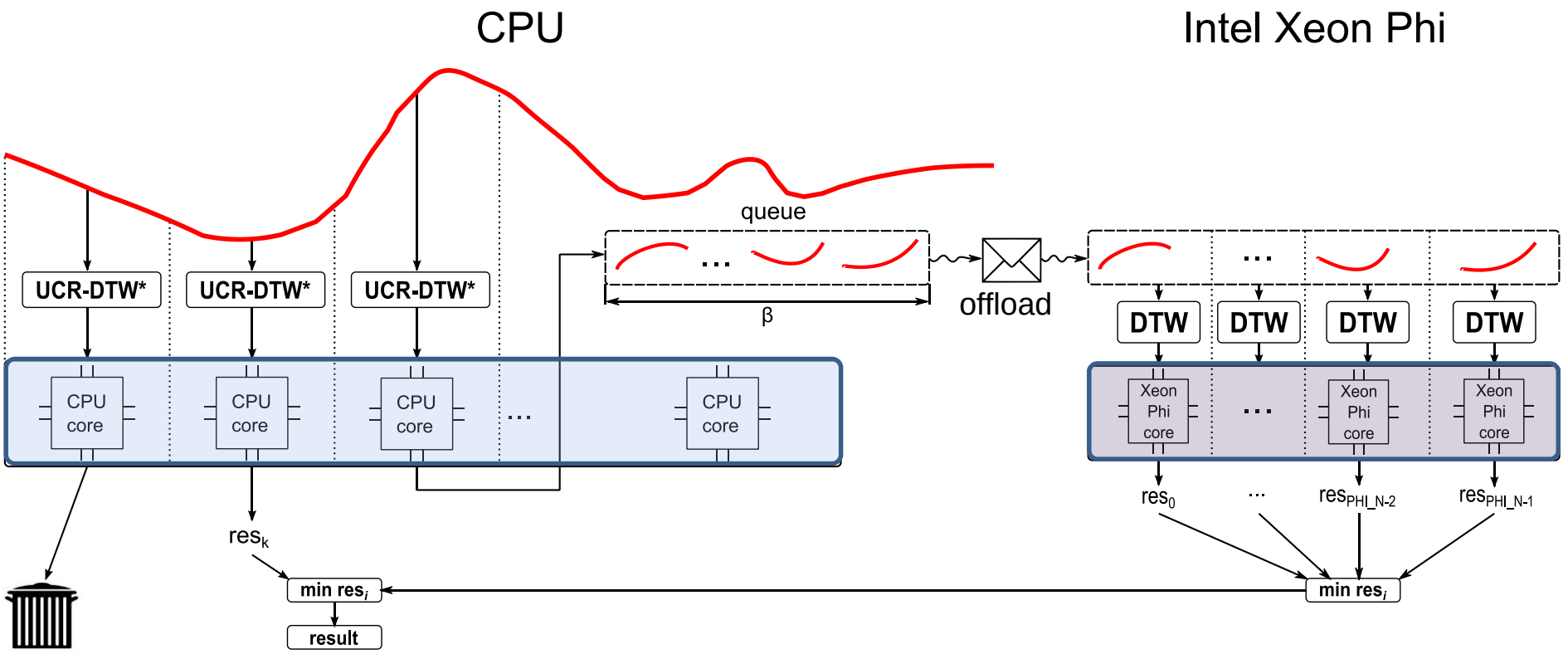
Результаты экспериментов, наивный алгоритм



LB_Kim	O(1)
LB_Keogh	O(n)
LB_KeoghEC	O(n)
DTW	O(n^2)

Синхронизация только при входе/выходе в offload-секцию

Параллельный алгоритм для Xeon Phi



Векторизация: до

```
double DTW(a: array [1..m], b: array [1..m], r: int) {
   cost := array [1..m]
   cost_prev := array [1..m]

   for i := 1 to m
      cost[i] = infinity
      cost_prev[i] = infinity

   cost_prev[1] = dist(a[1], b[1])

   for j := max(2, i-r) to min(m, i+r)
      cost_prev[j] := cost_prev[j-1] + dist(a[1], b[j])

   for i := 2 to m
      for j := max(1, i-r) to min(m, i+r)
         c := d(a[i], b[j])
         cost[j] := c + min(cost[j-1], cost_prev[j-1], cost_prev[j])
      swap(cost, cost_prev)

   return cost_prev[m]
}
```

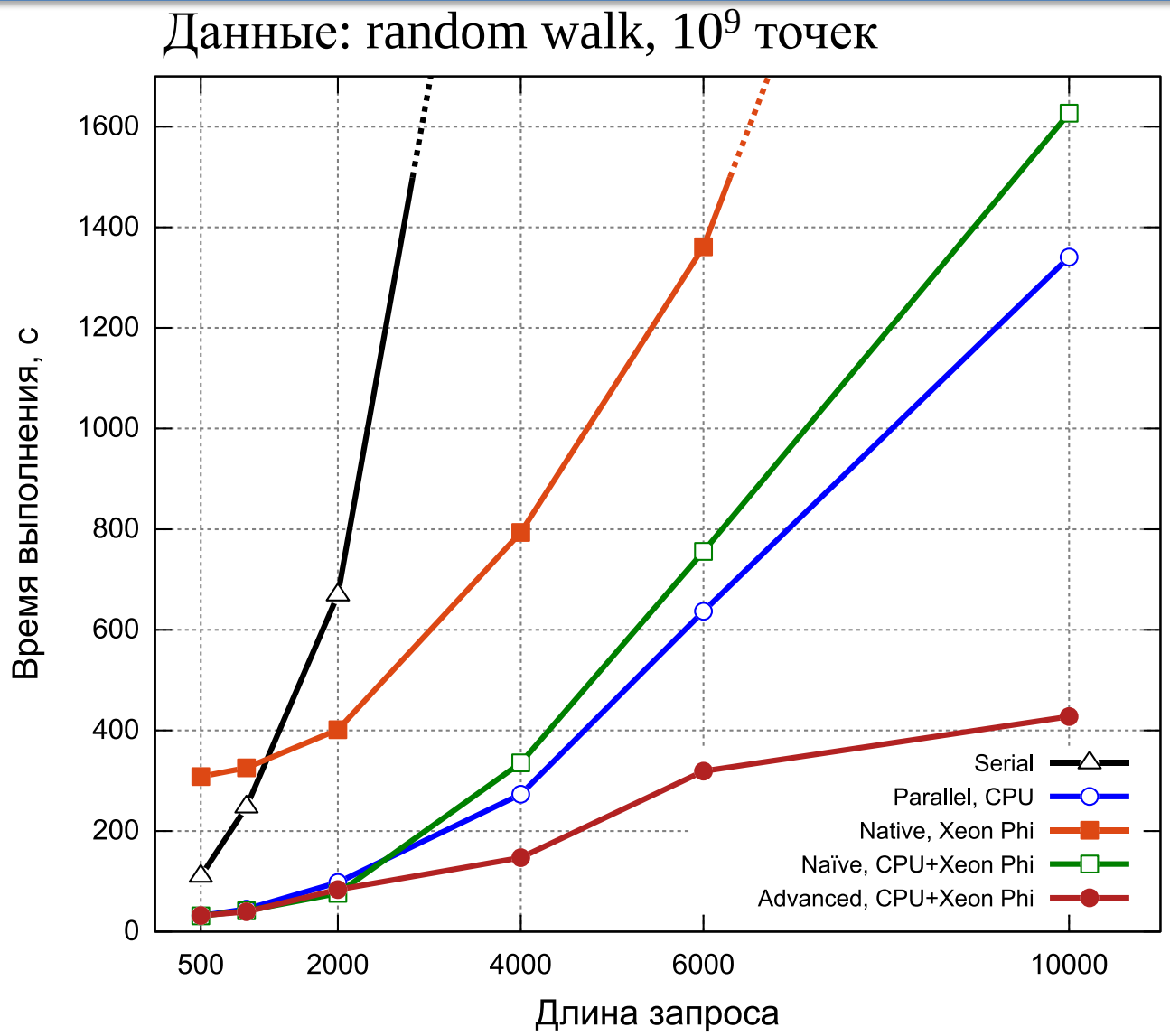
Векторизация: после

```
double DTW(a: array [1..m], b: array [1..m], r: int) {
    cost := array [1..m]
    cost_prev := array [1..m]
    for i := 1 to m
        cost[i] = infinity
        cost_prev[i] = infinity
    cost_prev[1] = dist(a[1], b[1])
    for j := max(2, i-r) to min(m, i+r)
        cost_prev[j] := cost_prev[j-1] + dist(a[1], b[j])
    for i := 2 to m
        for j := max(1, i-r) to min(m, i+r)
            cost[j] = min(cost_prev[j-1], cost_prev[j])
        for j := max(1, i-r) to min(m, i+r)
            c := dist(a[i], b[j])
            cost[j] := c + min(cost[j-1], cost[j])
        swap(cost, cost_prev)
    return cost_prev[m]
}
```

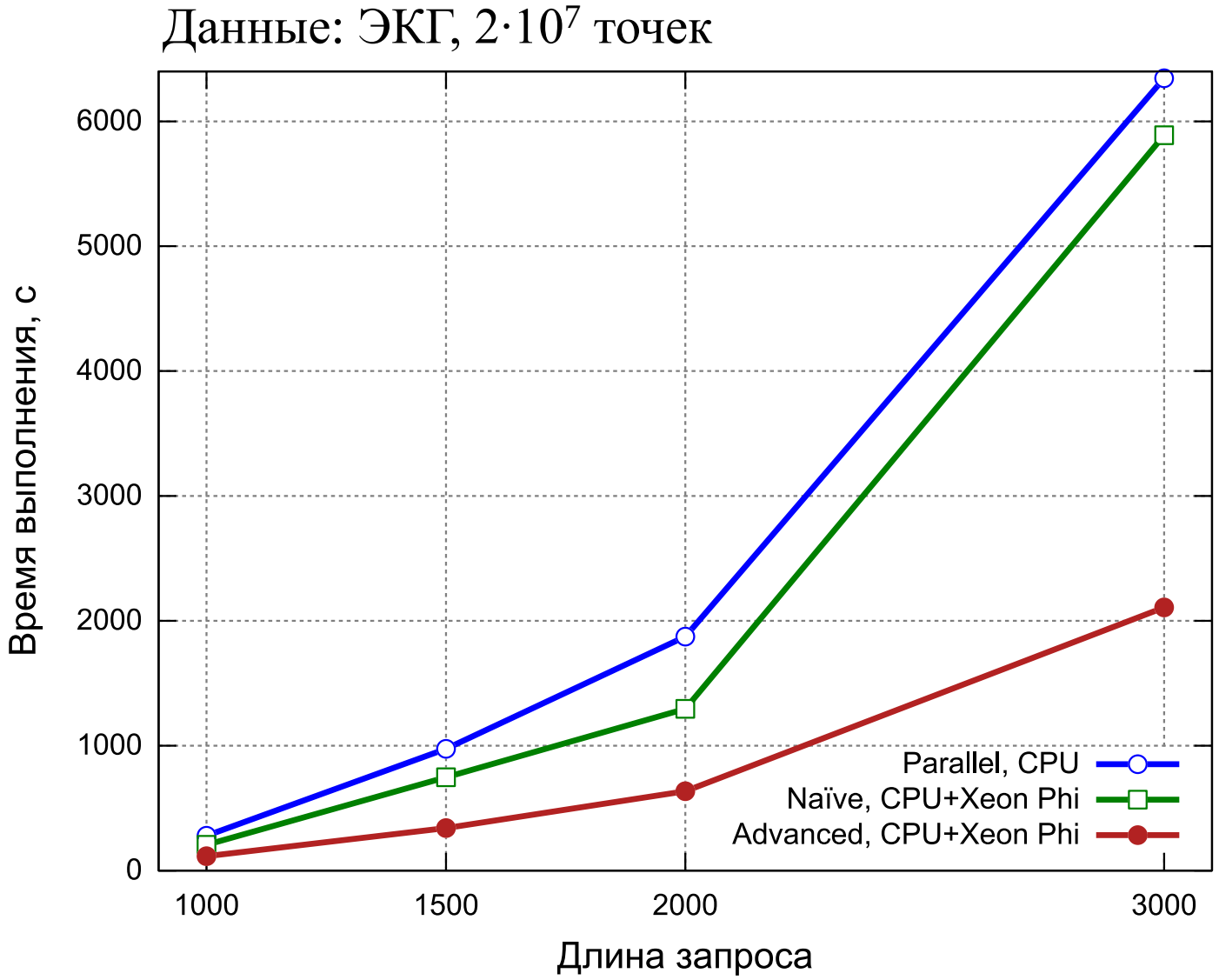
Эксперименты

- Аппаратная платформа
  - Процессор
    - Intel Xeon X5680
    - 6 ядер по 3.33 GHz
    - 0.371 Тфлопс
  - Сопроцессор
    - Intel Xeon Phi SE10X
    - 61 ядро по 1.1 GHz
    - 1.076 Тфлопс
- Данные
  - Синтетические
    - random walk, 10^9 точек данных
  - Реальные
    - ЭКГ, $2 \cdot 10^7$ точек данных (22 час. при частоте дискретизации 250 Гц)

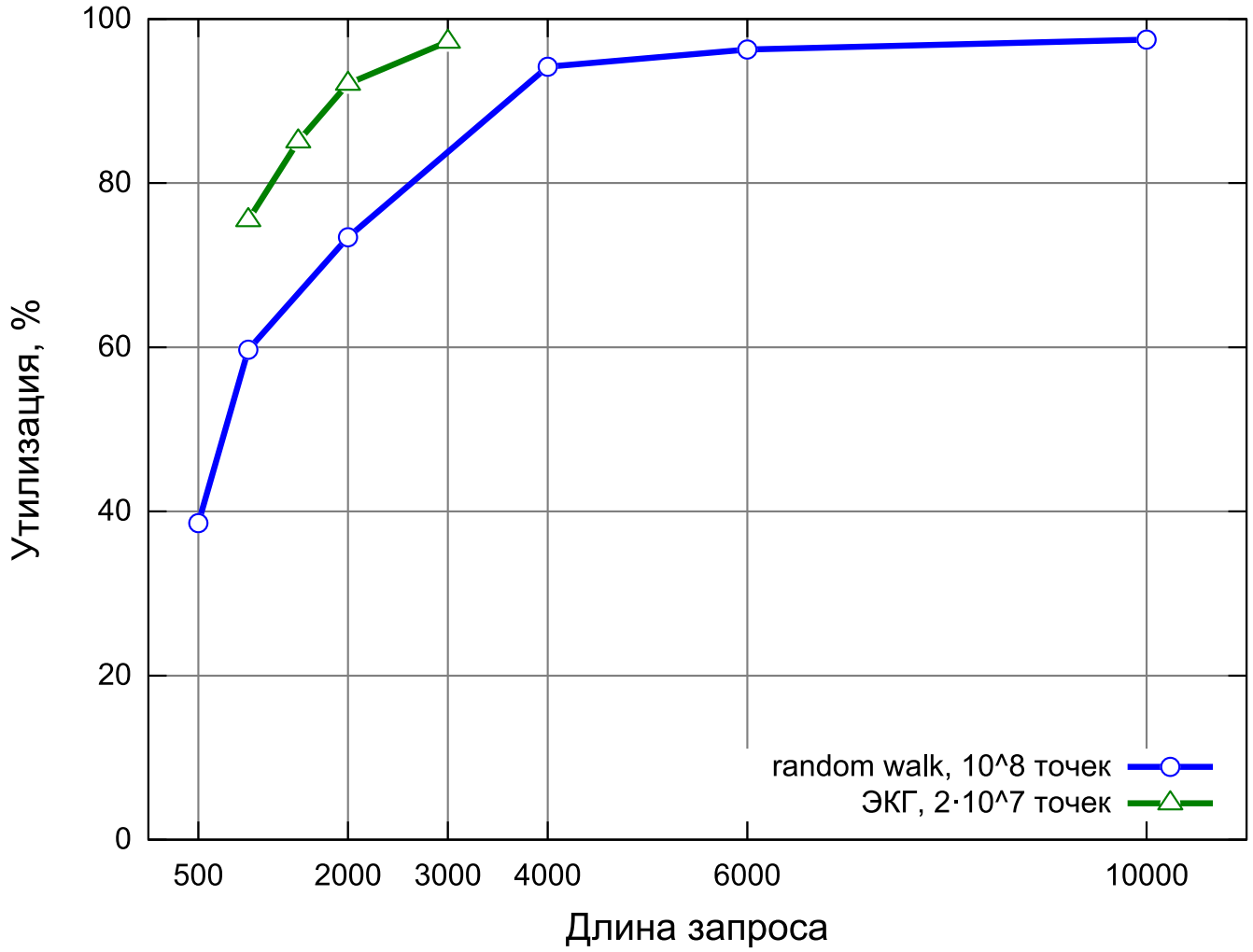
Эксперименты: синтетич. данные



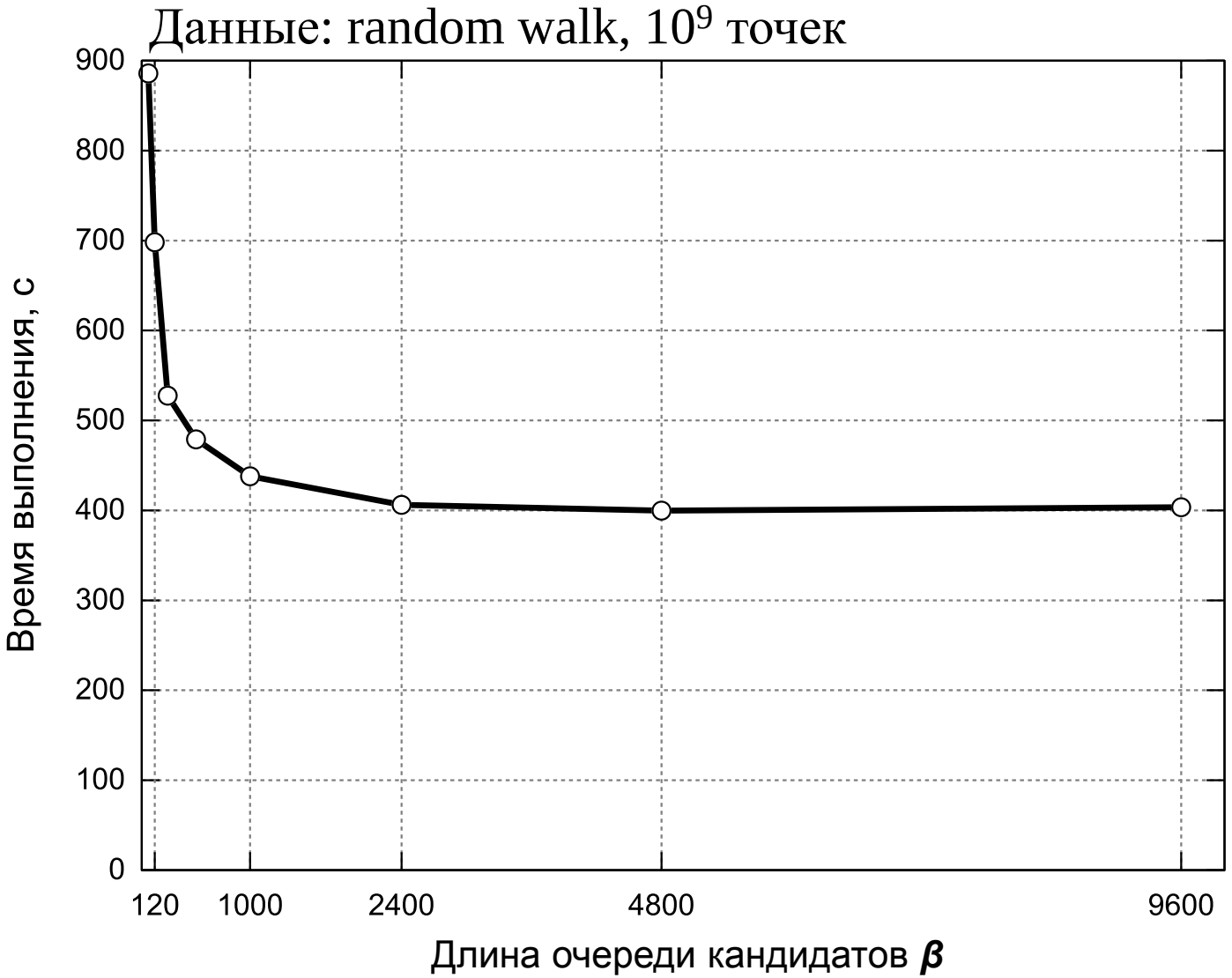
Эксперименты: ЭКГ



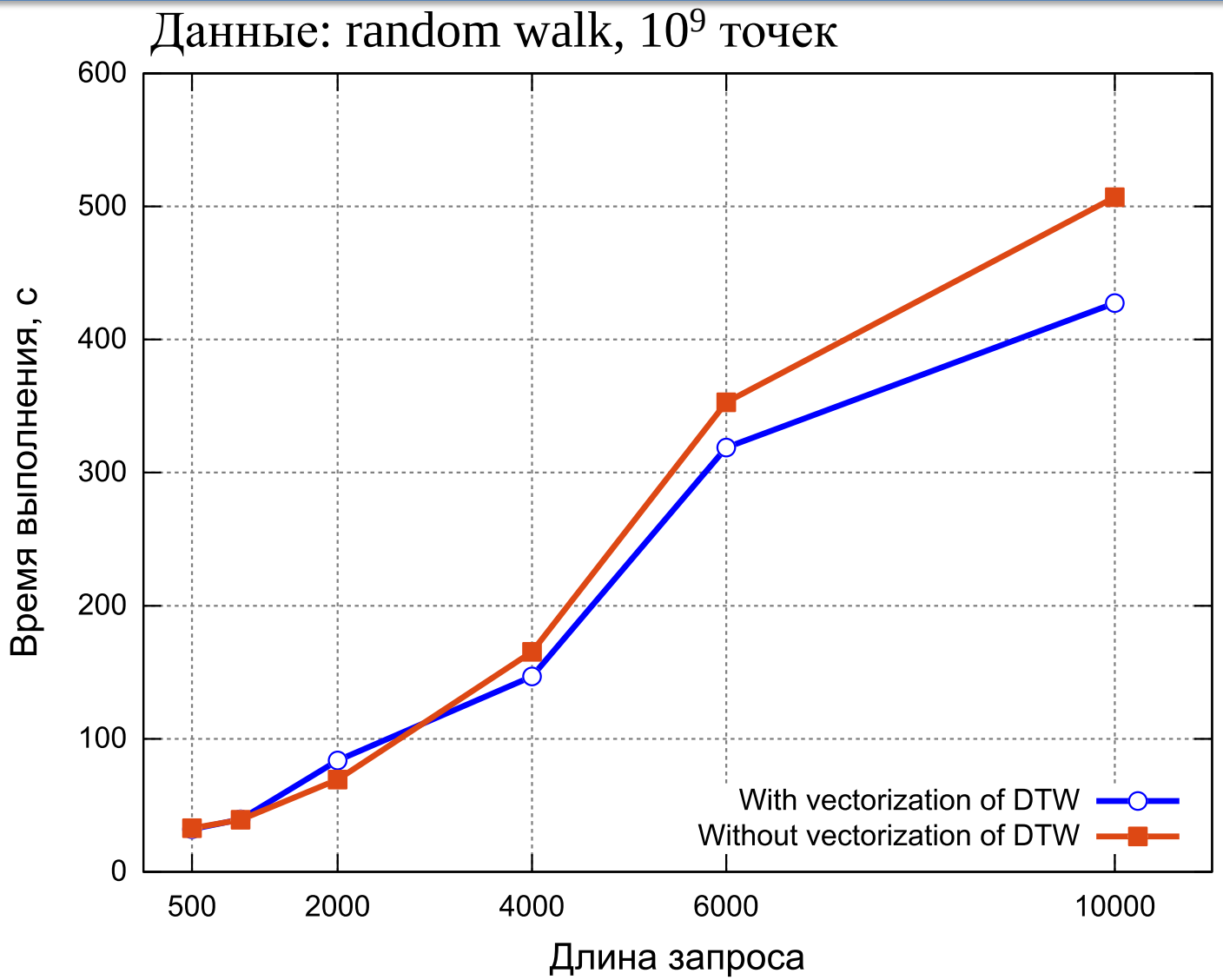
Эксперименты: утилизация Phi



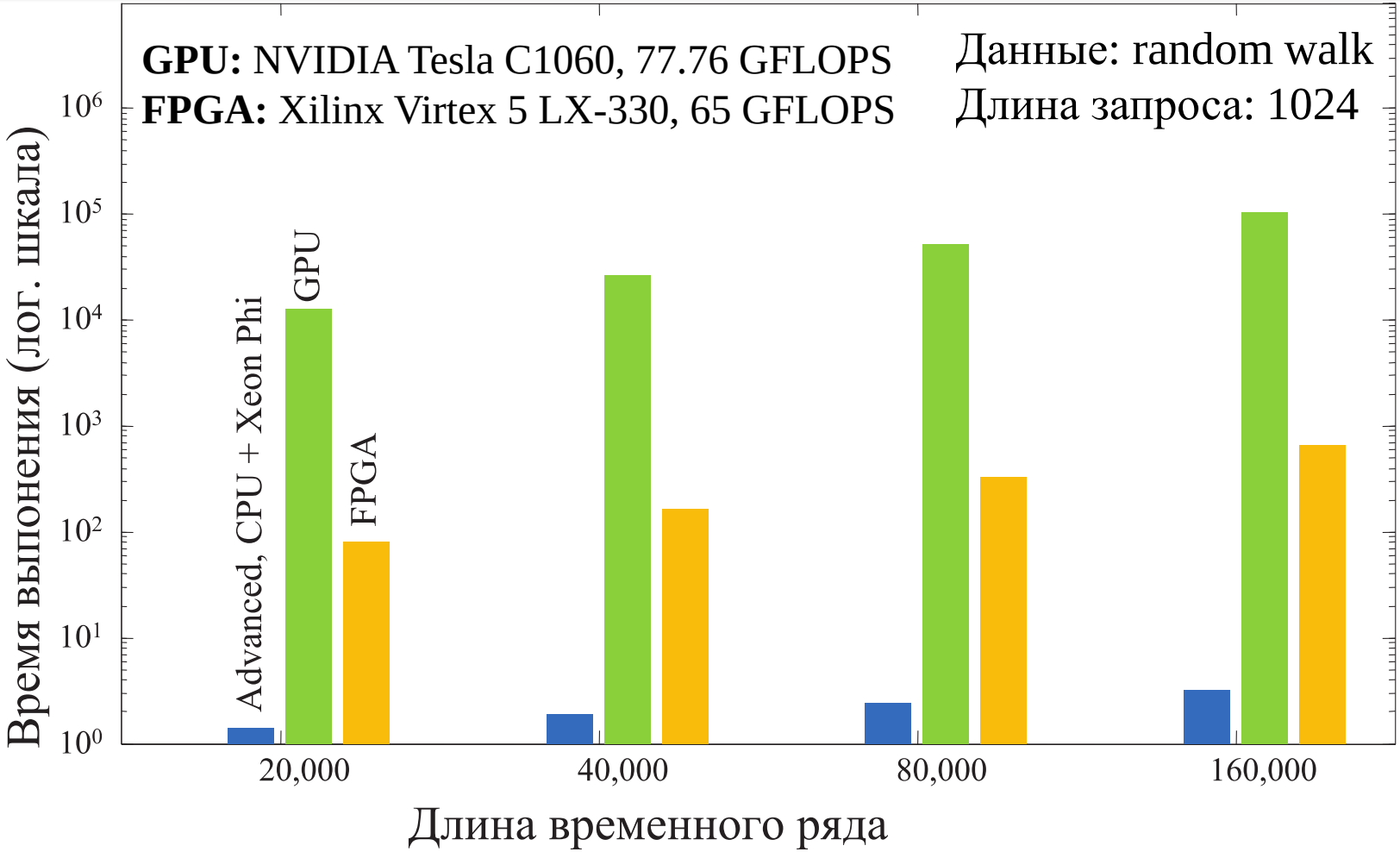
Эксперименты: длина очереди



Эксперименты: векторизация



Эксперименты: сравнение



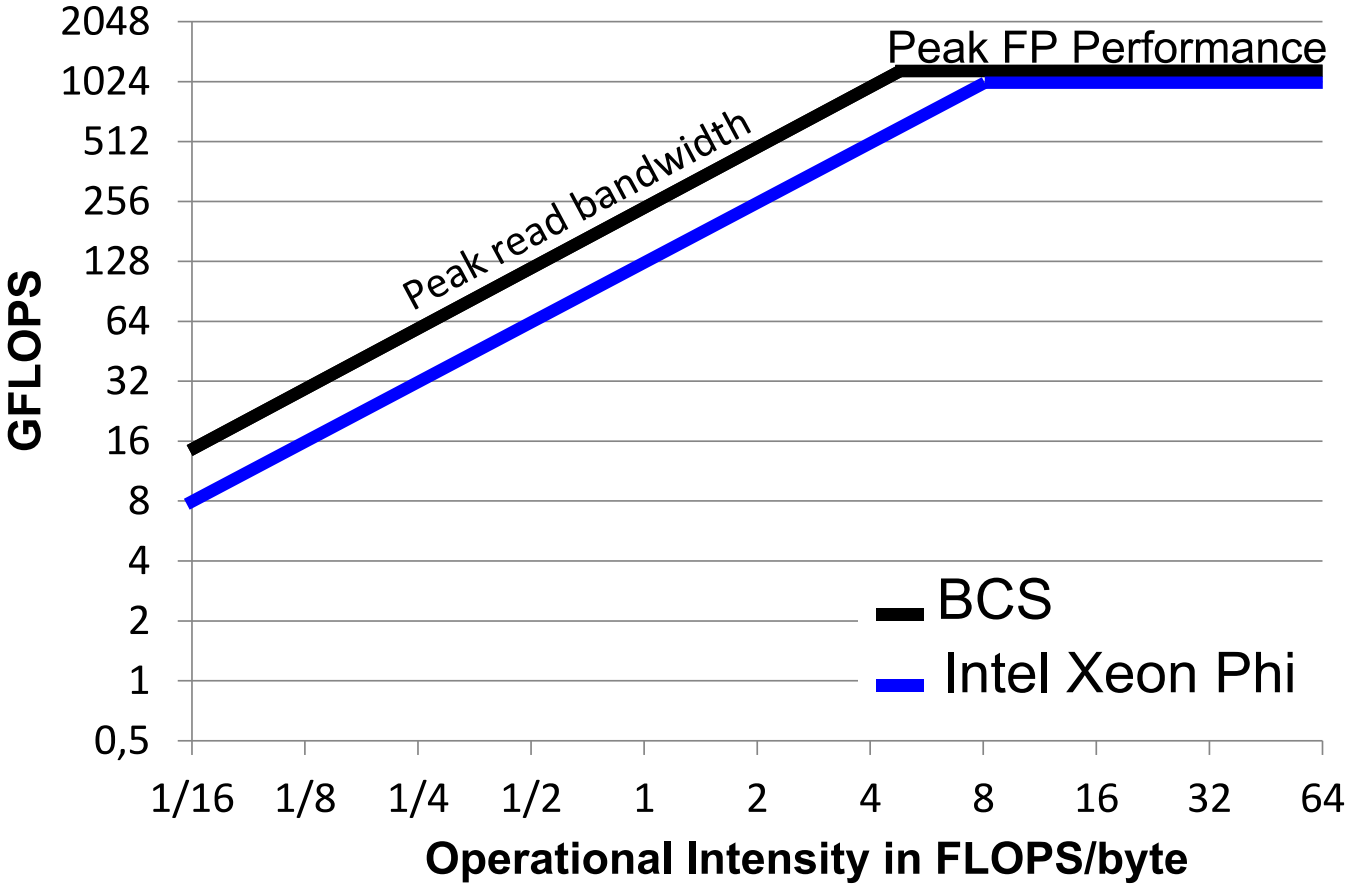
Sart D., et al. Accelerating Dynamic Time Warping Subsequence Search with GPUs and FPGAs // The 10th IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13-17 December, 2010. IEEE, 2010. P. 1001-1006.

Заключение

- Новый параллельный алгоритм поиска похожих подпоследовательностей временного ряда для сопроцессоров Intel Xeon Phi.
- Эксперименты: высокая эффективность алгоритма при большой длине запроса.
- Будущие исследования: кластерная система с узлами на базе сопроцессоров Intel Xeon Phi.

- Спасибо за внимание
  - Вопросы?
  - Михаил Леонидович Цымблер
    mzym@susu.ru

Roofline

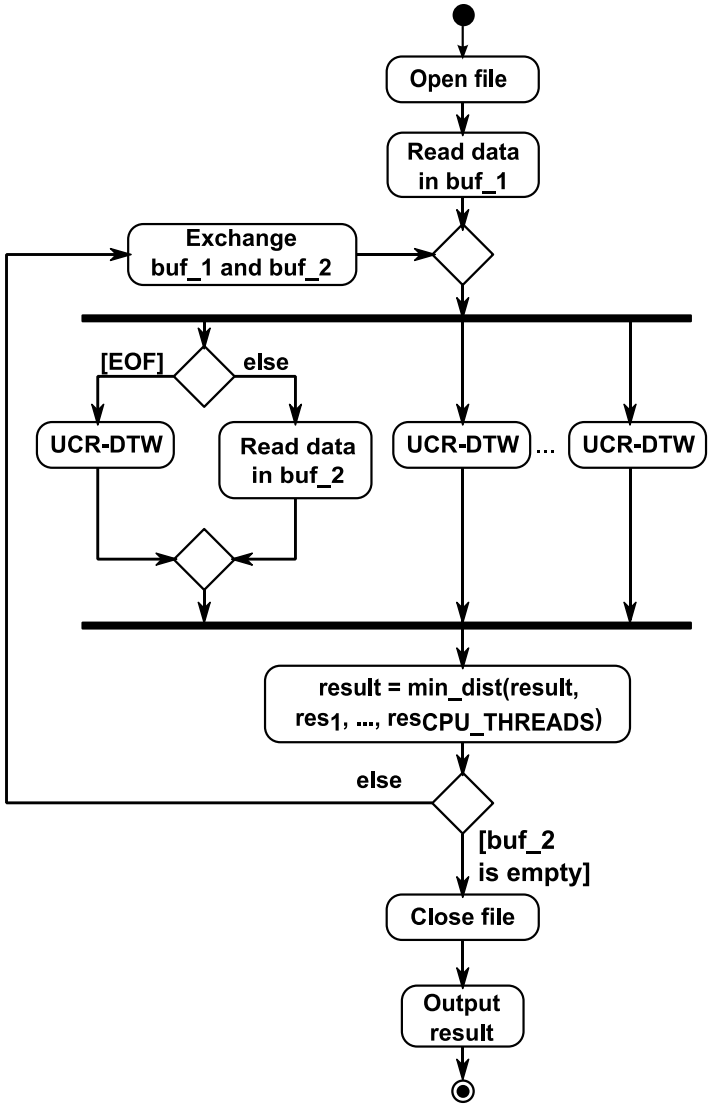


BCS: 4 bullx s6010 boards
Board: 4 Intel Xeon X7550 (Nehalem-EX), 64 GB

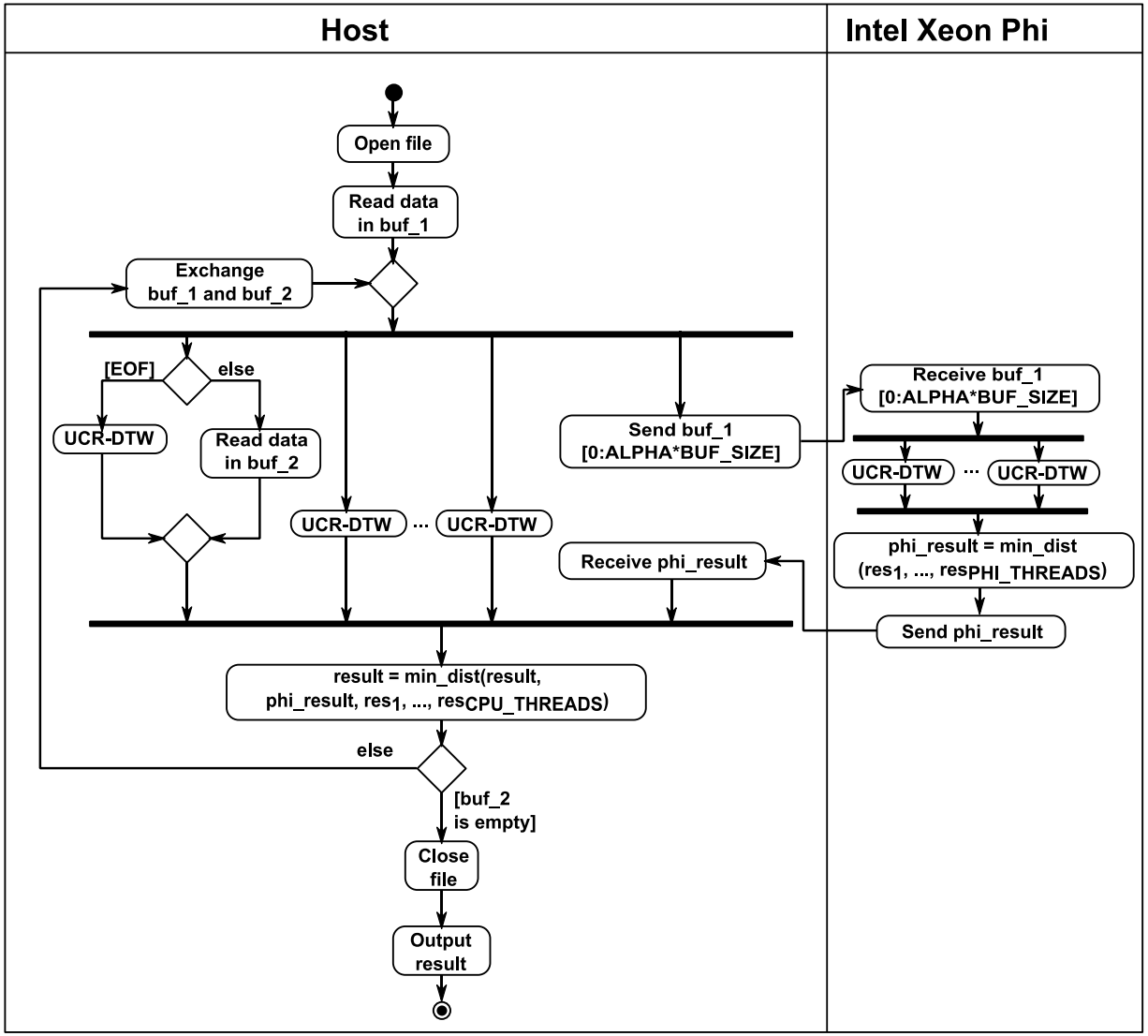
Cramer T., et al. OpenMP Programming on Intel Xeon Phi Coprocessors: An Early Performance Comparison // Many-core Applications Research Community Symposium, Aachen, Germany, 29-30 November, 2012. RWTH Aachen University, 2012. P. 38-44.

Параллельный алгоритм для CPU

- OpenMP



Наивный алгоритм для Xeon Phi



Параллельный алгоритм для Xeon Phi