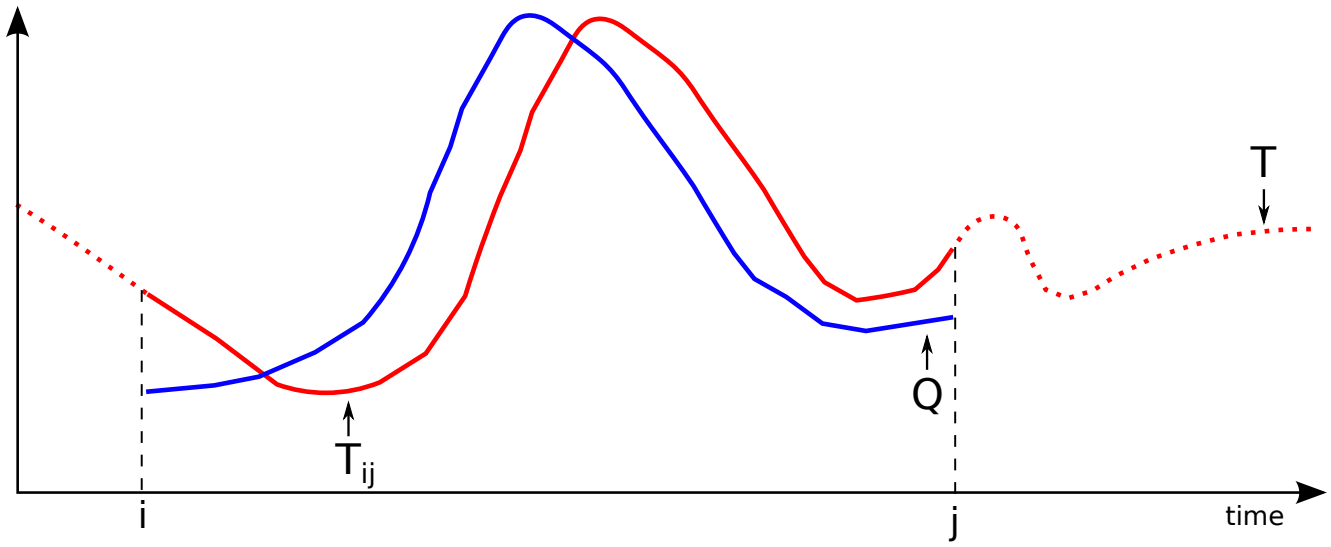
Нахождение похожих подпоследовательностей временного ряда с помощью многоядерного сопроцессора Intel Xeon Phi

Александр Вячеславович Мовчан

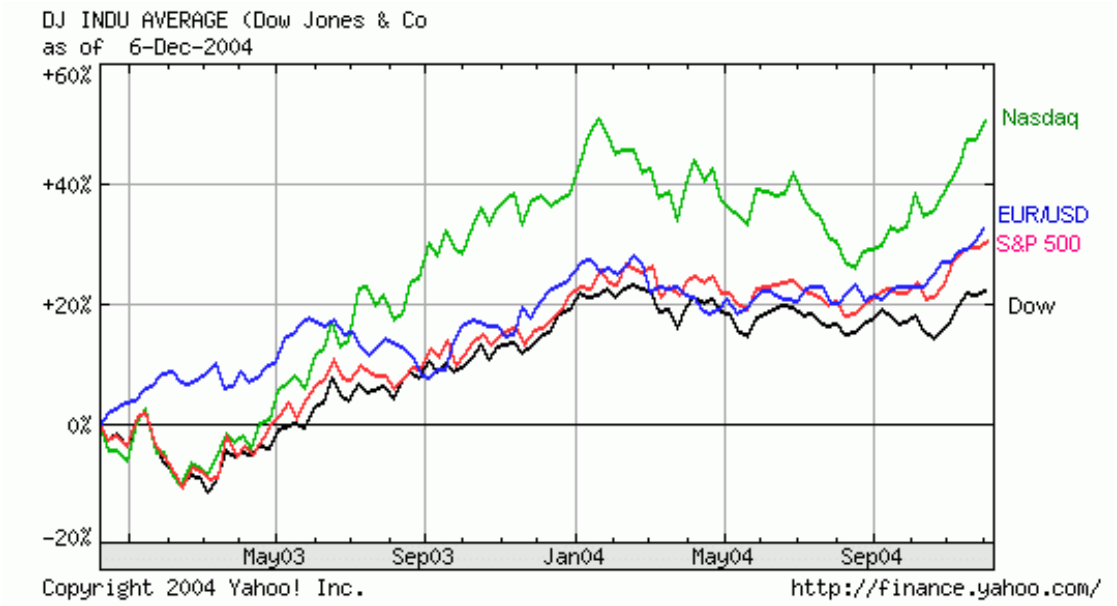
Научный руководитель: **М.Л. Цымблер**

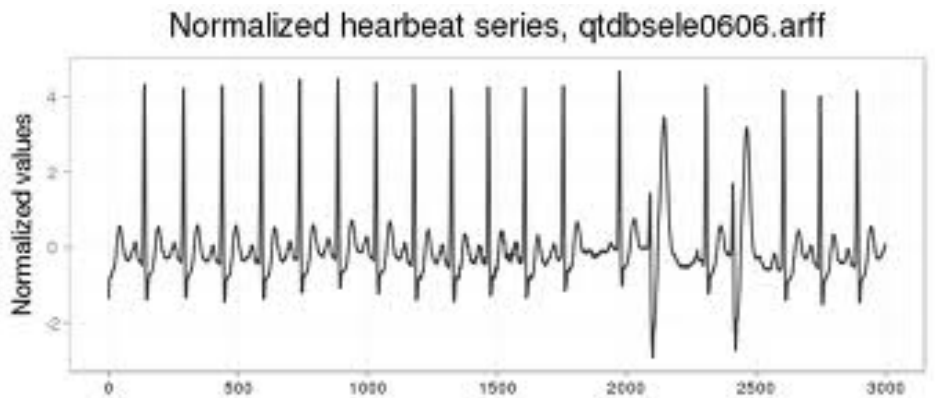
Основные обозначения



- *Временной ряд (time series)* T — упорядоченная последовательность $t_1, t_2, ..., t_N$ (где N — длина последовательности) вещественных значений, каждое из которых ассоциировано с отметкой времени.
- *Подпоследовательность (subsequence)* T_{ij} временного ряда T представляет собой непрерывное подмножество T, начинающееся с позиции i и заканчивающееся в позиции j.
- *Запрос (query)* Q — последовательность, длина которой меньше N.

Применение временных рядов





Поиск похожей подпоследовательности

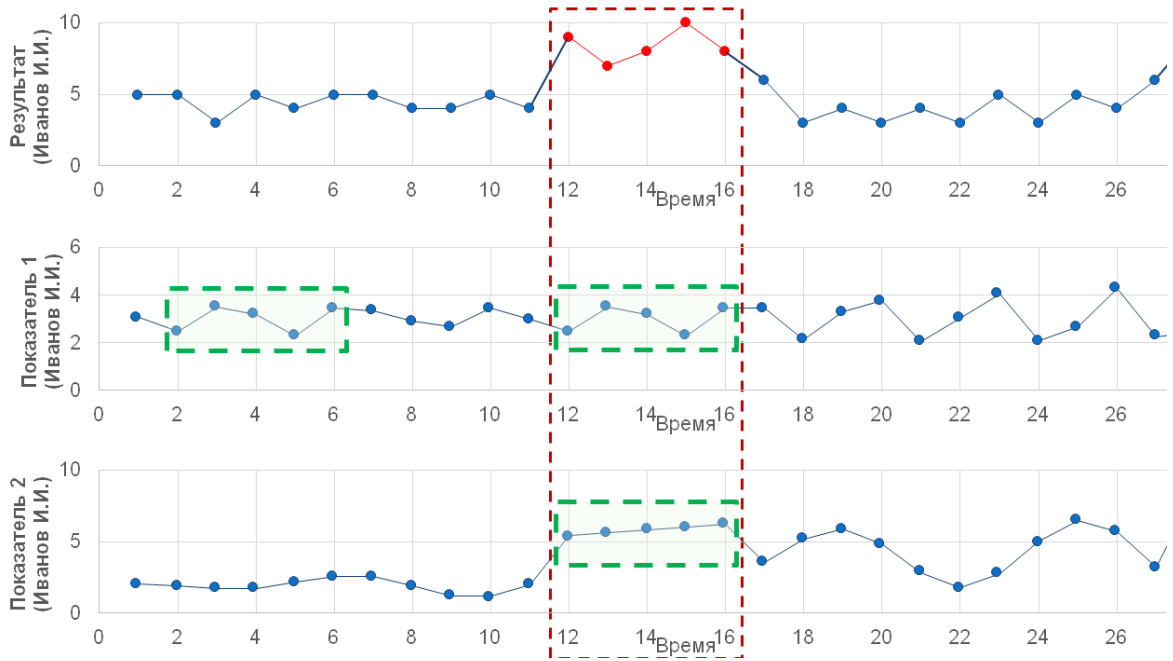
Best-match search: результатом является подпоследовательность $T_{ij} \in T$, для которой выполнены следующие условия:
$\forall T_{mn}$

- $|T_{mn}| = |T_{ij}| = |Q|$
- $D(T_{ij}, Q) < D(T_{mn}, Q)$

Local-best-match search: результатом является множество подпоследовательностей $\{T_{ij} \in T\}$, для которых выполнены следующие условия:

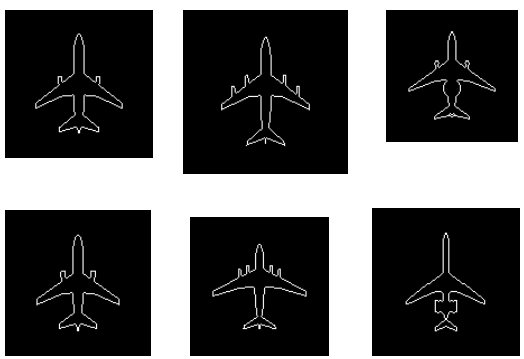
- $|T_{ij}| = |Q|$
- $D(T_{ij}, Q) < \mathcal{E}$
- $D(T_{ij}, Q) < D(T_{i-1j-1}, Q)$
- $D(T_{ij}, Q) < D(T_{i+1j+1}, Q)$

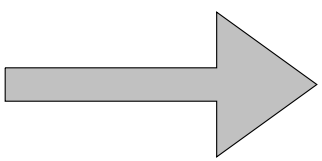
Проект MedMining

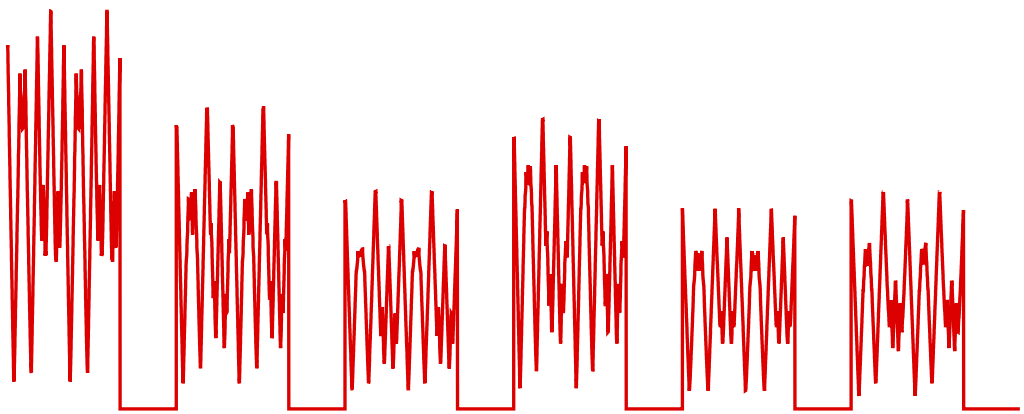


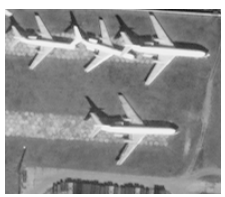
Интеллектуальный анализ данных физиологических исследований профессиональных спортсменов

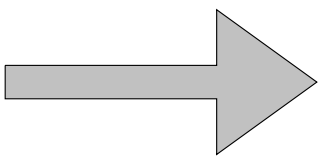
Классификация контуров

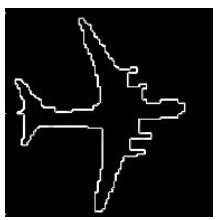


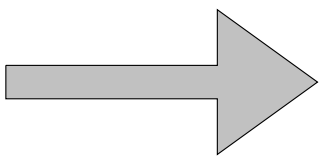


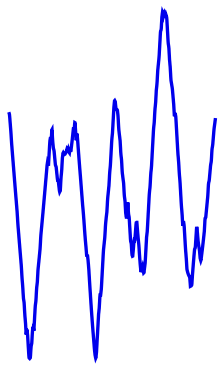






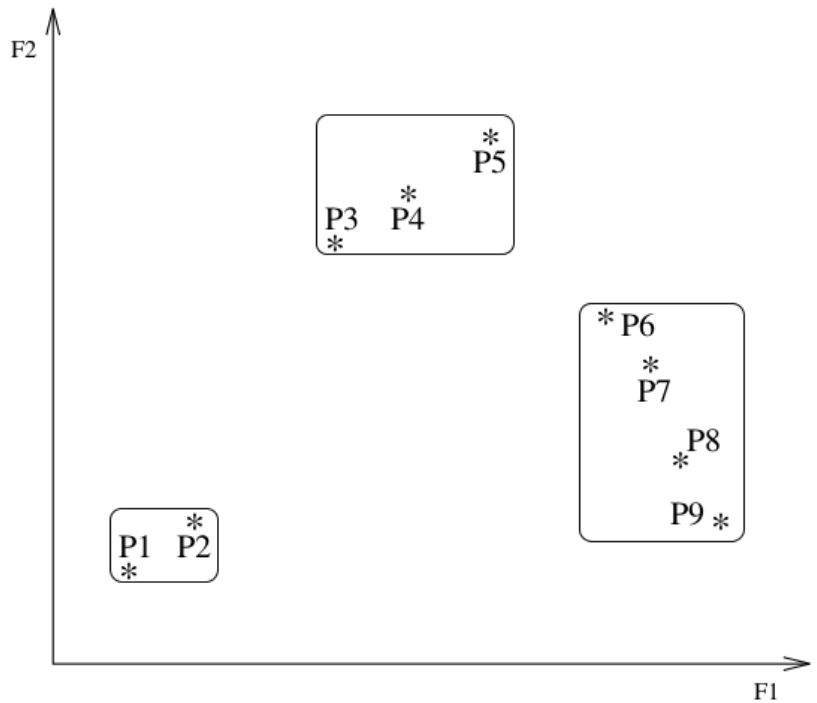






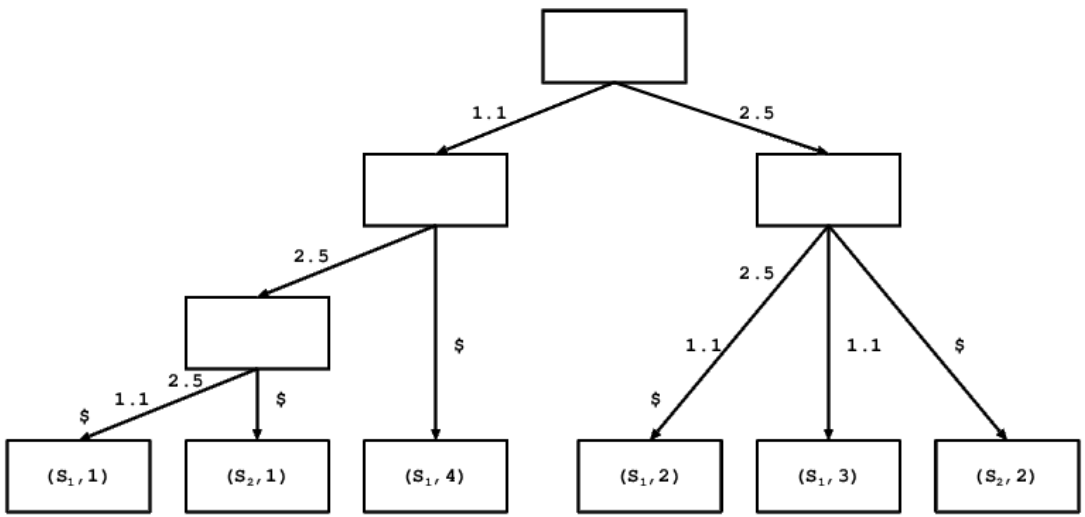
Методы поиска похожих подпоследовательностей

- Построение индекса
  - Многомерные прямоугольники
    - для каждой подпоследовательности вычисляется преобразование Фурье
    - строится индекс по этим данным
    - выполняется поиск по индексу с помощью многомерных прямоугольников



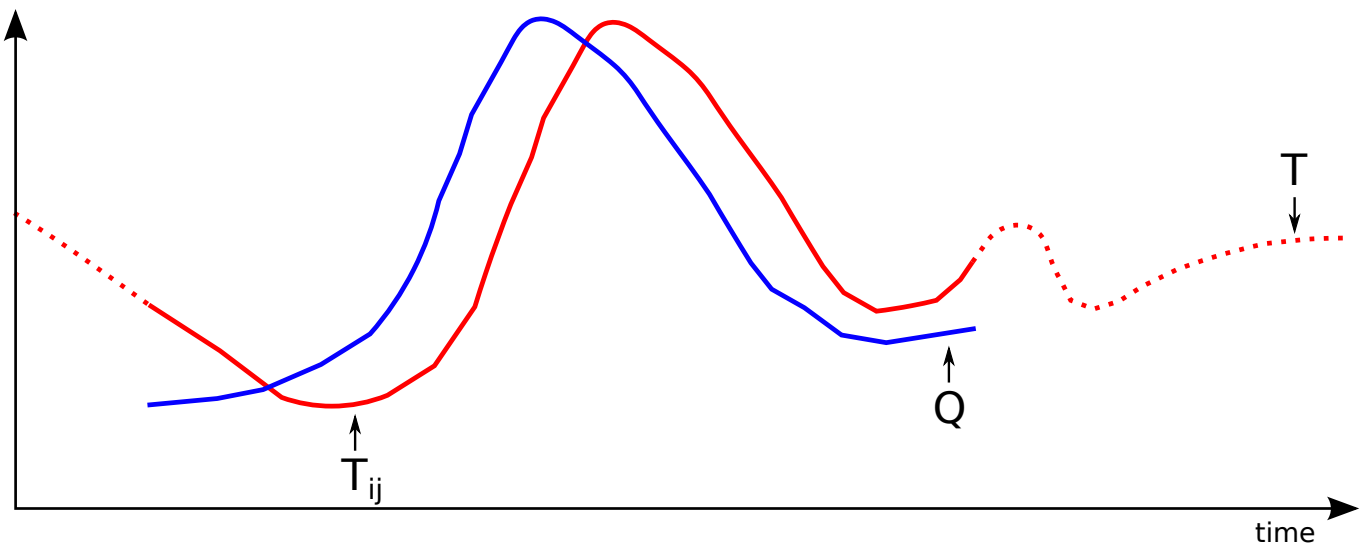
Методы поиска похожих подпоследовательностей

- Построение индекса
  - Суффиксные деревья
    - по временному ряду строится суффиксное дерево (индекс)
    - выполняется запрос к суффиксному дереву
  - Префикс запроса
    - индекс строится как в случае многомерных прямоугольников
    - выполняется поиск по индексу по префиксу запроса



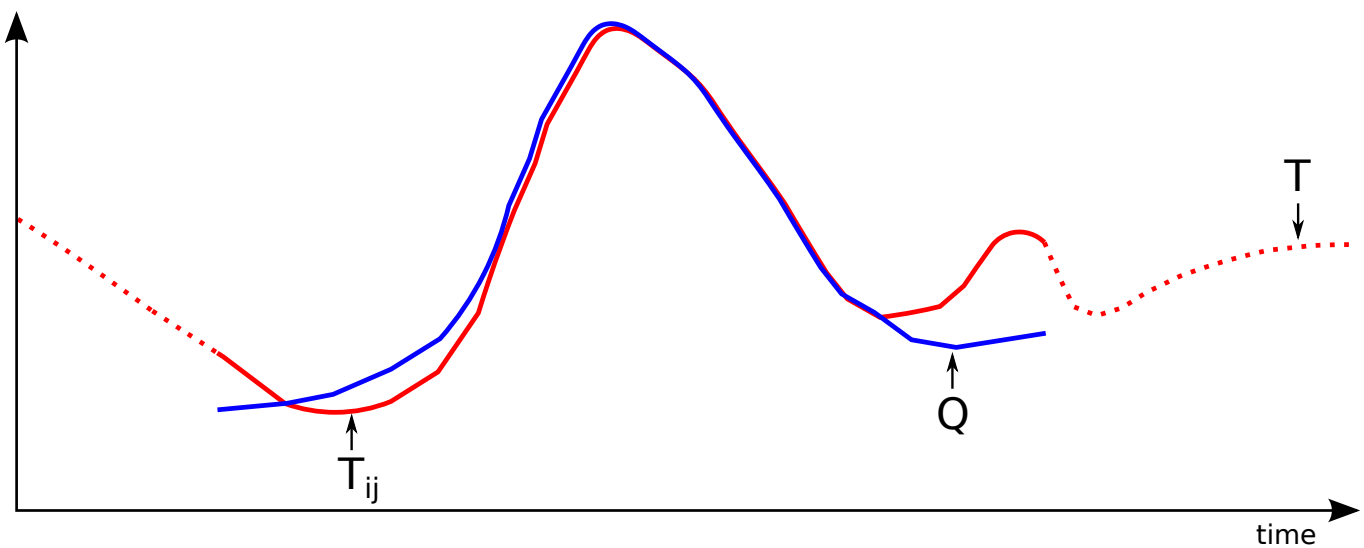
Методы поиска похожих подпоследовательностей

- Последовательное сравнение
  - для каждой подпоследовательности осуществляется сравнение



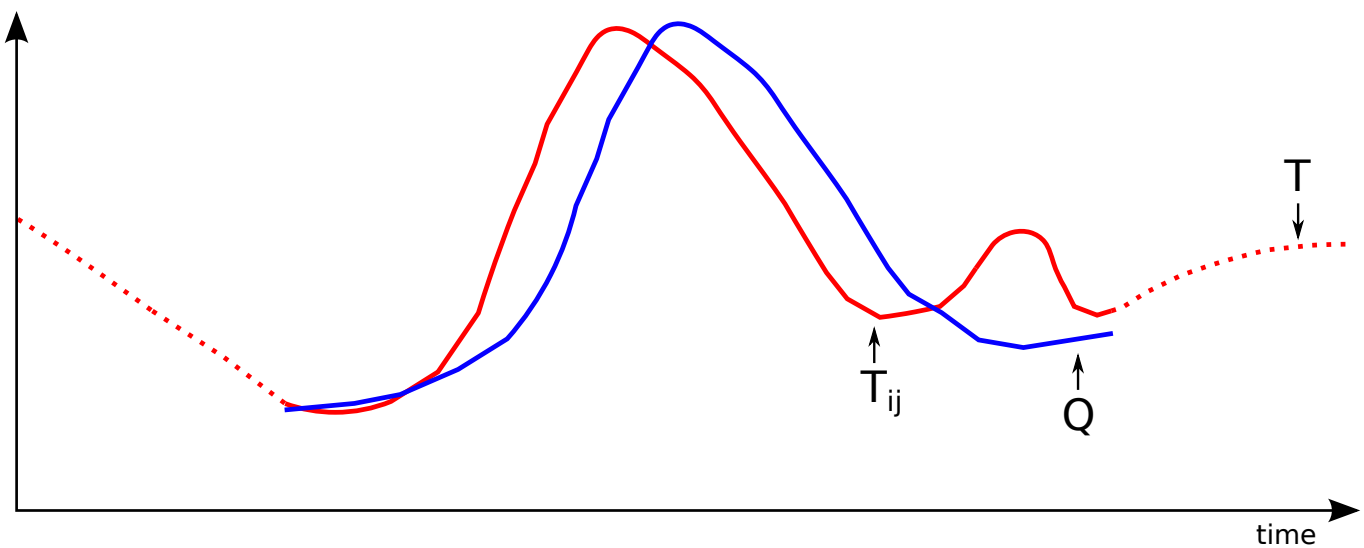
Методы поиска похожих подпоследовательностей

- Последовательное сравнение
  - для каждой подпоследовательности осуществляется сравнение



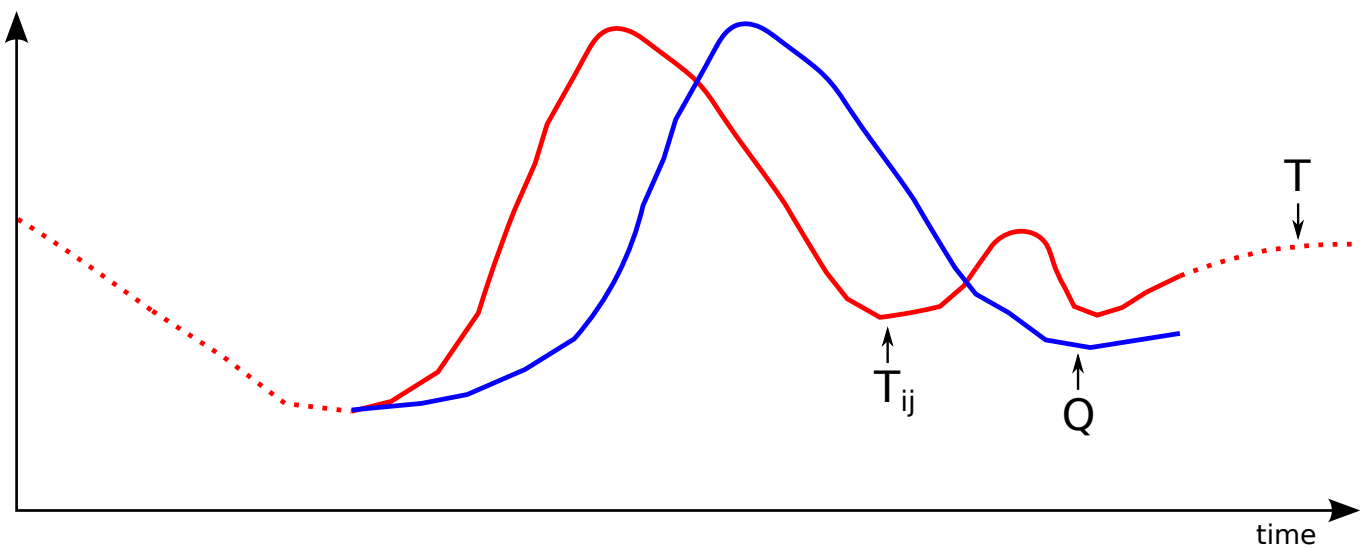
Методы поиска похожих подпоследовательностей

- Последовательное сравнение
  - для каждой подпоследовательности осуществляется сравнение



Методы поиска похожих подпоследовательностей

- Последовательное сравнение
  - для каждой подпоследовательности осуществляется сравнение



Обзор работ по поиску похожих подпоследовательностей на основе DTW

- Последовательные алгоритмы
  - Исследования, подтверждающие преимущество DTW перед другими мерами схожести в различных предметных областях.
    - Ding et al. Querying and mining of time series data: experimental comparison of representations and distance measures // PVLDB, 2008.
    - Fu et al. Scaling and time warping in time series querying // VLDB, 2008.
  - Повторное использование результатов вычислений при подсчете DTW.
    - Sakurai et al. Stream monitoring under the time warping distance // ICDE, 2007.
  - Построение индекса для поиска подпоследовательности, похожей на запрос с заранее заданной длиной.
    - Lim et al. Using multiple indexes for efficient subsequence matching in time-series databases // DASFAA, 2006.
  - Построения множества индексов для для поиска подпоследовательности, похожей на один из запросов с заранее заданными длинами.
    - Keogh et al. Supporting exact indexing of arbitrarily rotated shapes and periodic time series under euclidean and warping distance measures // VLDB, 2009.

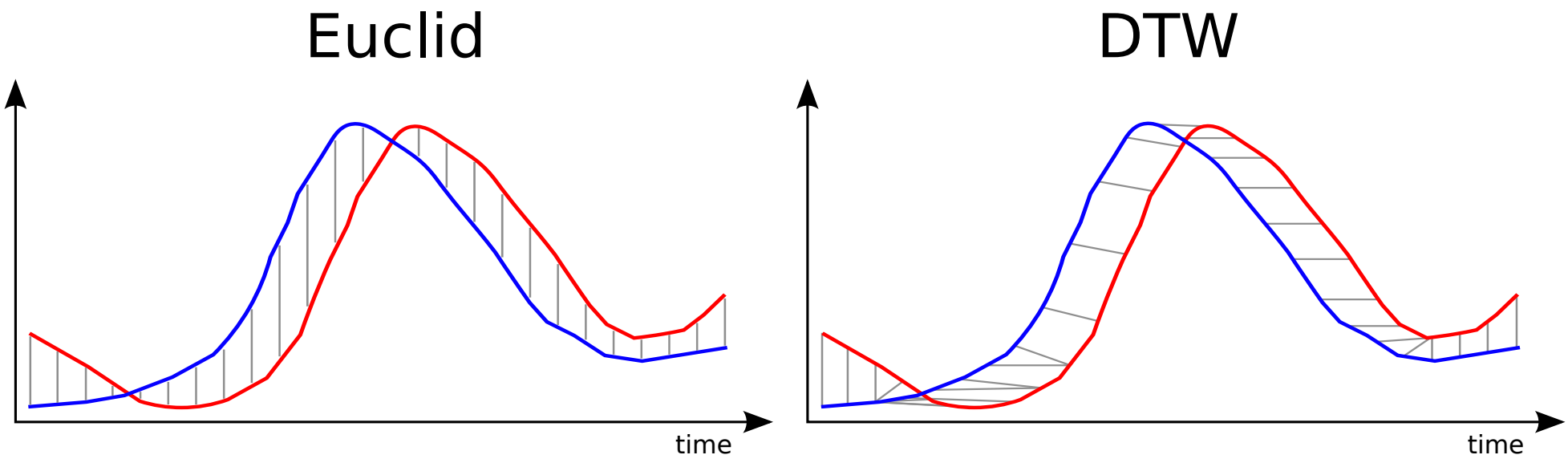
Обзор работ по поиску похожих подпоследовательностей на основе DTW

- Последовательные алгоритмы
  - Отбрасывание заведомо непохожих подпоследовательностей на основе предварительных оценок расстояния от запроса до подпоследовательности (lower bounding).
    - Fu et al. Scaling and time warping in time series querying // VLDB, 2005.
  - UCR-DTW, наиболее быстрый в настоящее время алгоритм на основе lower bounding).
    - Rakthanmanon et al. Searching and mining trillions of time series subsequences under dynamic time warping // KDD, 2012.

Обзор работ по поиску похожих подпоследовательностей на основе DTW

- Параллельные алгоритмы
  - Многопоточная реализация.
    - Takahashi et al. A parallelized data stream processing system using dynamic time warping distance // CISIS, 2009.
  - Реализация для кластера на базе Intel Xeon.
    - Sharanyan et al. Implementing the dynamic time warping algorithm in multithreaded environments for real time and unsupervised pattern discovery // ICCCT, 2011.
  - Реализации для GPU и FPGA (по тому же принципу, что и Srikanthan et al.).
    - Sart et al. Accelerating dynamic time warping subsequence search with GPUs and FPGAs // ICDM, 2010.
  - Реализация для GPU (параллельное формирование матрицы трансформации, вычисление DTW последовательно).
    - Zhang et al. Fast spoken query detection using lower-bound dynamic time warping on graphical processing units // ICASSP, 2012.
  - Улучшенная реализация для FPGA.
    - Wang et al. Accelerating subsequence similarity search based on dynamic time warping distance with FPGA // ACM/SIGDA, 2013.
  - Реализация для Intel Xeon Phi.
    - Movchan et al. Accelerating time series subsequence matching on the Intel Xeon Phi many-core coprocessor // MIPRO, 2015.

Динамическая трансформация шкалы времени

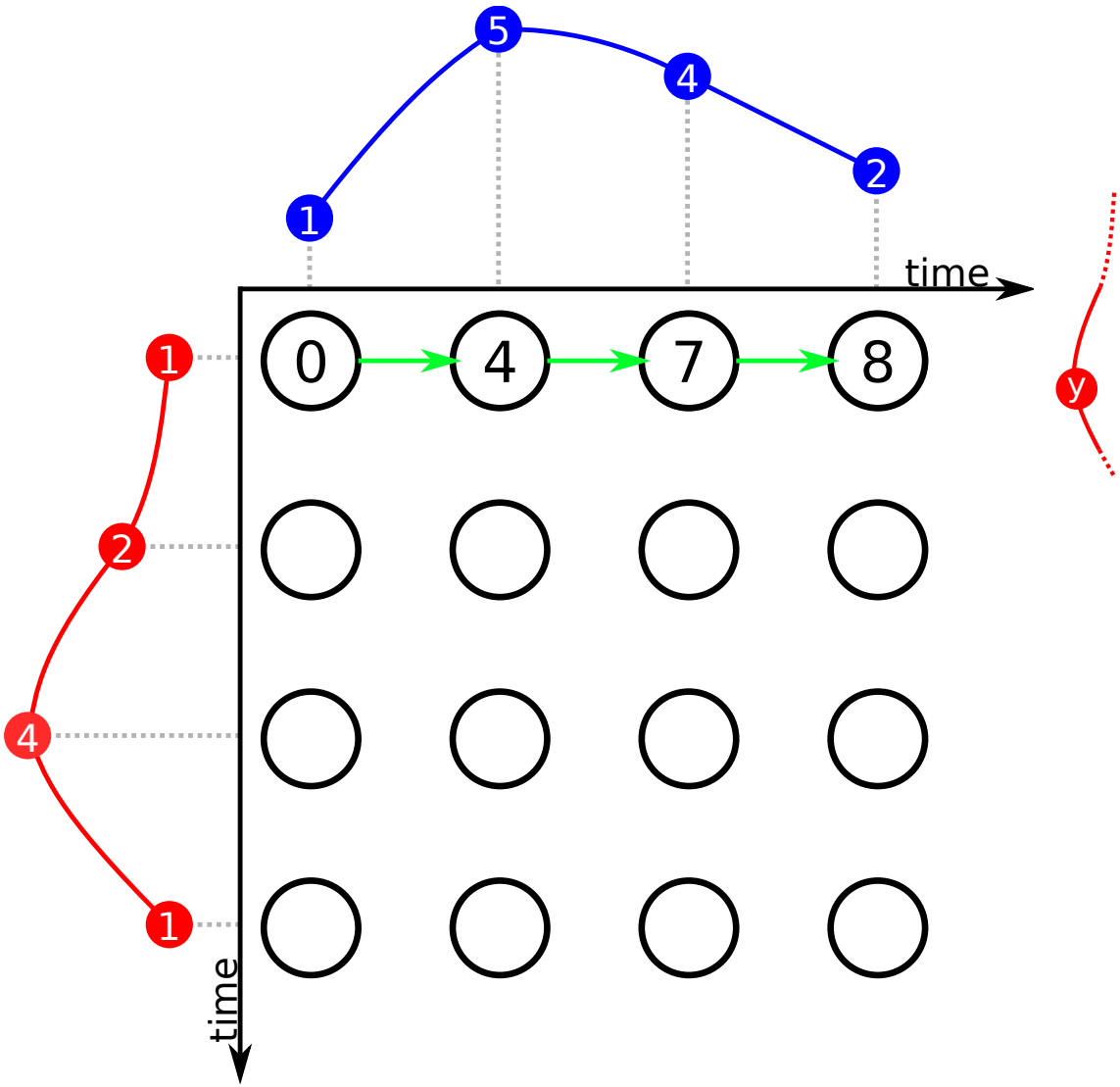


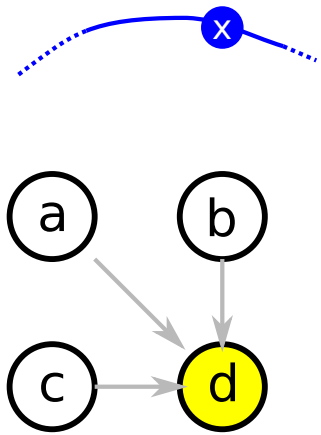
$$DTW(X,Y) = d(N,N),$$

$$d(i,j) = |x_i - y_j| + min \begin{cases} d(i-1,j) \\ d(i,j-1) \\ d(i-1,j-1), \end{cases}$$

$$d(0,0) = 0; d(i,0) = d(0,j) = \infty; i = 1,2,\ldots,N; j = 1,2,\ldots,N.$$

Динамическая трансформация шкалы времени

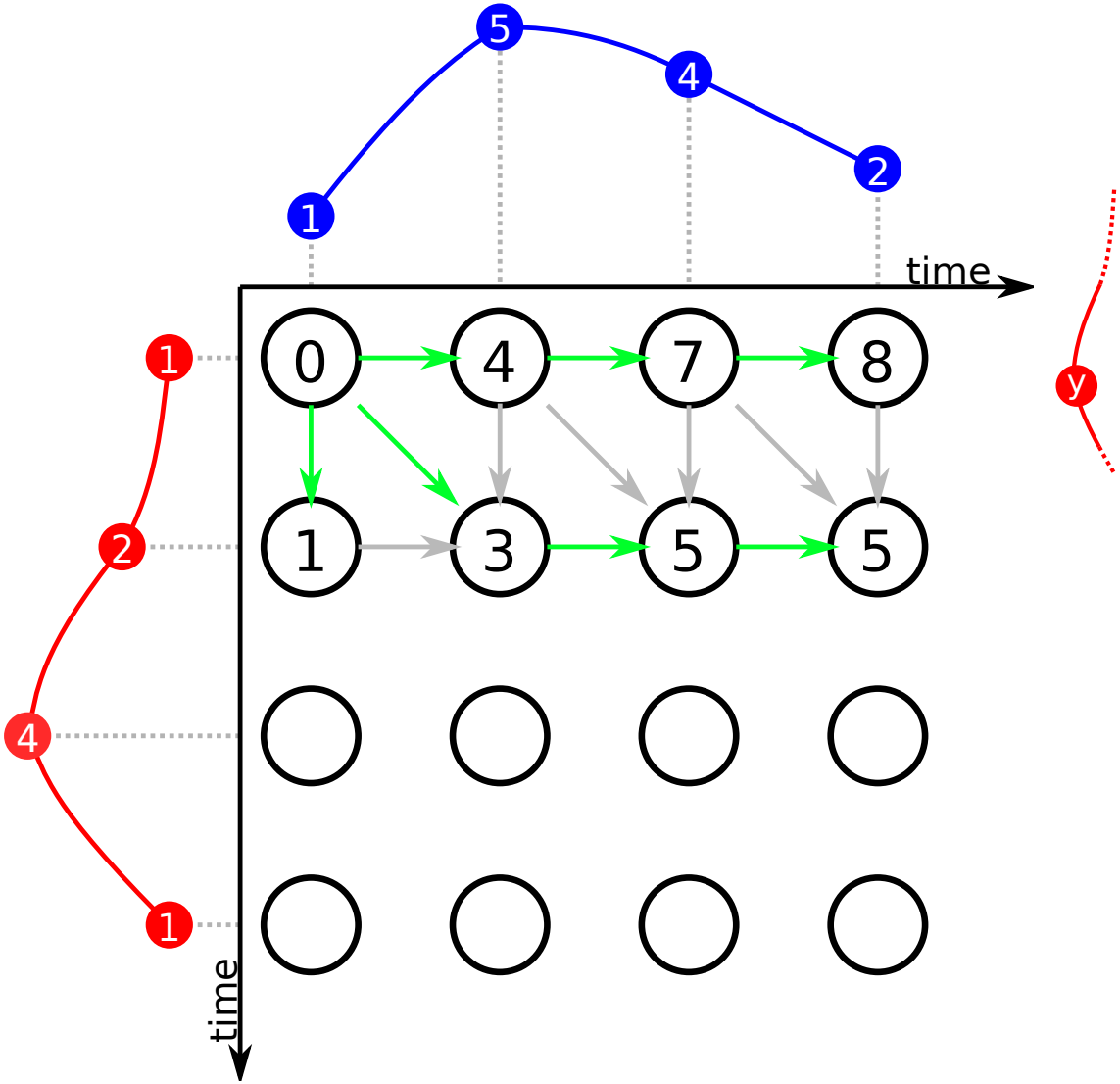


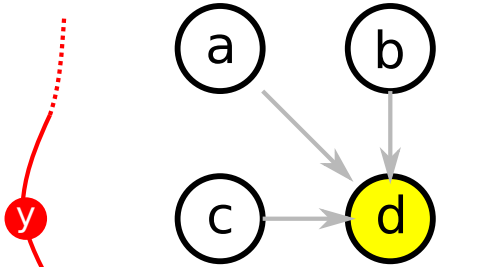


d = cost + min(a, b, c)

cost = |x - y|

Динамическая трансформация шкалы времени

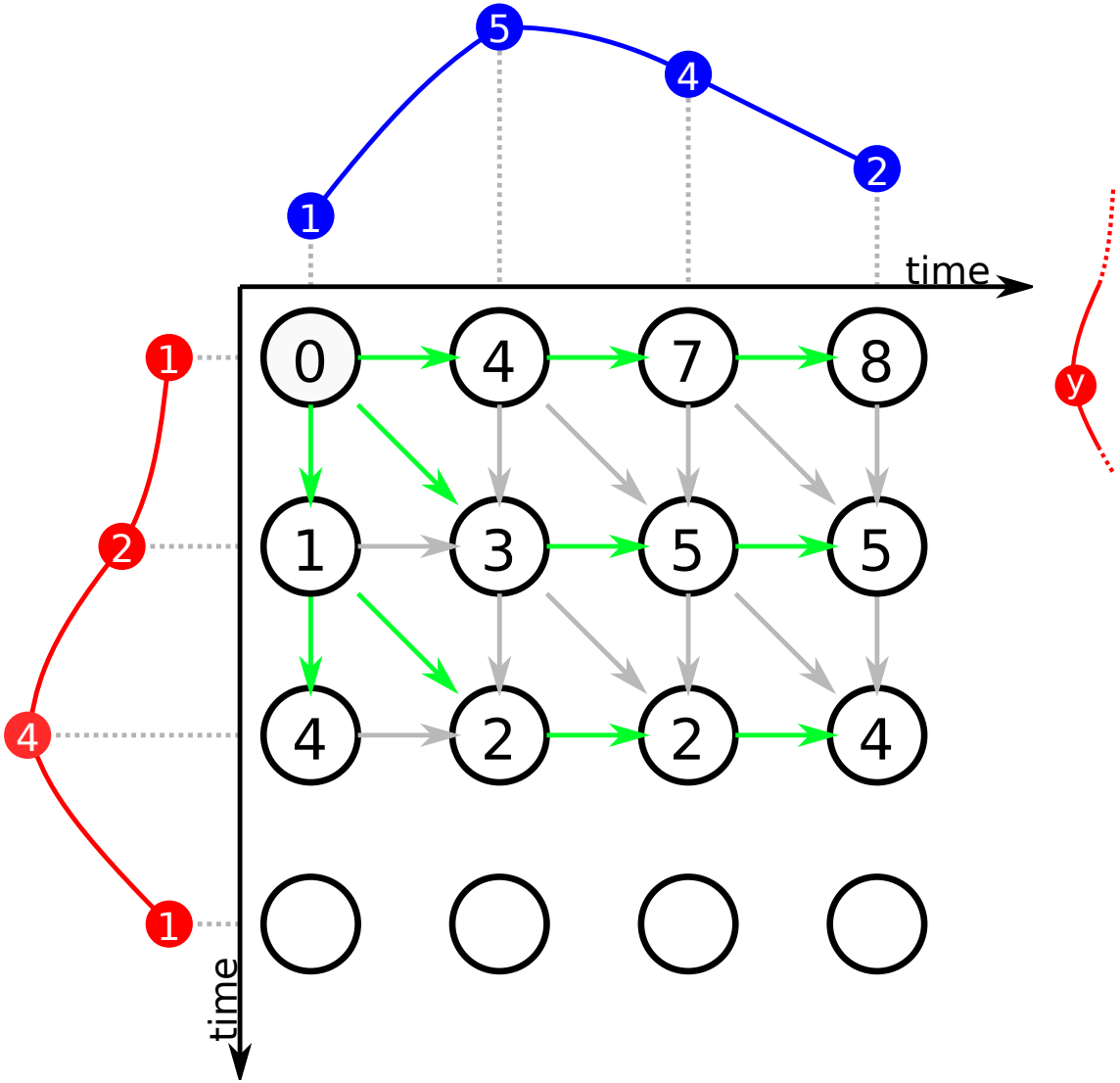


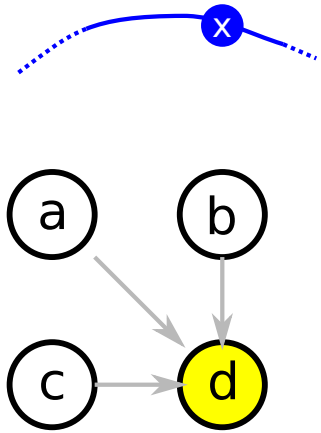


d = cost + min(a, b, c)

cost = |x - y|

Динамическая трансформация шкалы времени

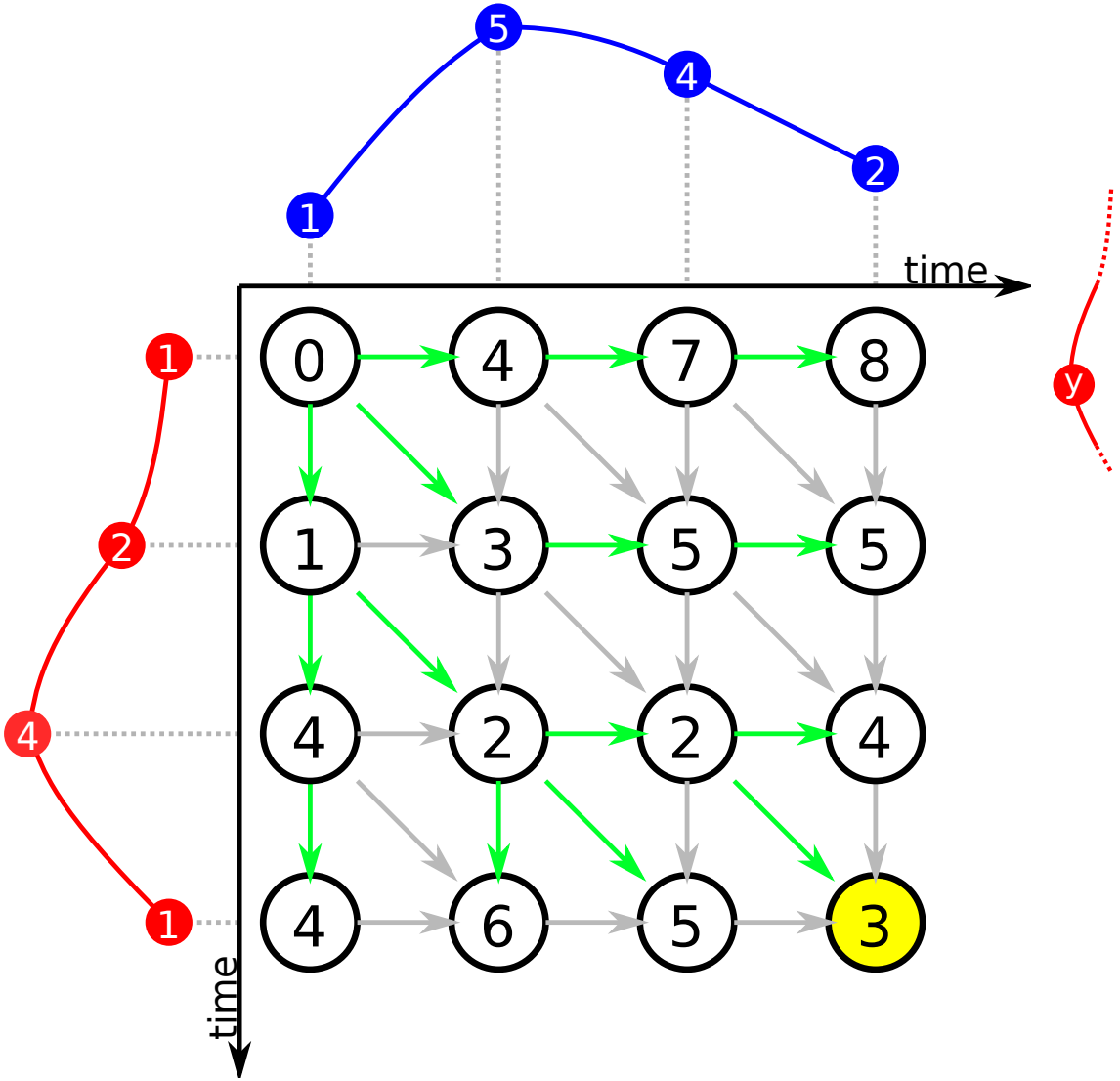


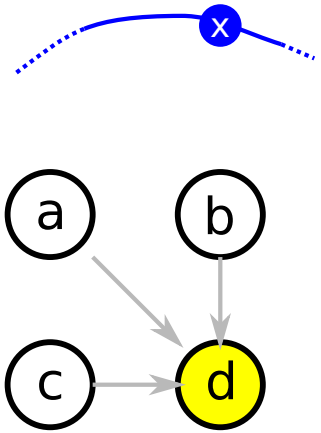


d = cost + min(a, b, c)

cost = |x - y|

Динамическая трансформация шкалы времени





d = cost + min(a, b, c)
cost = |x - y|

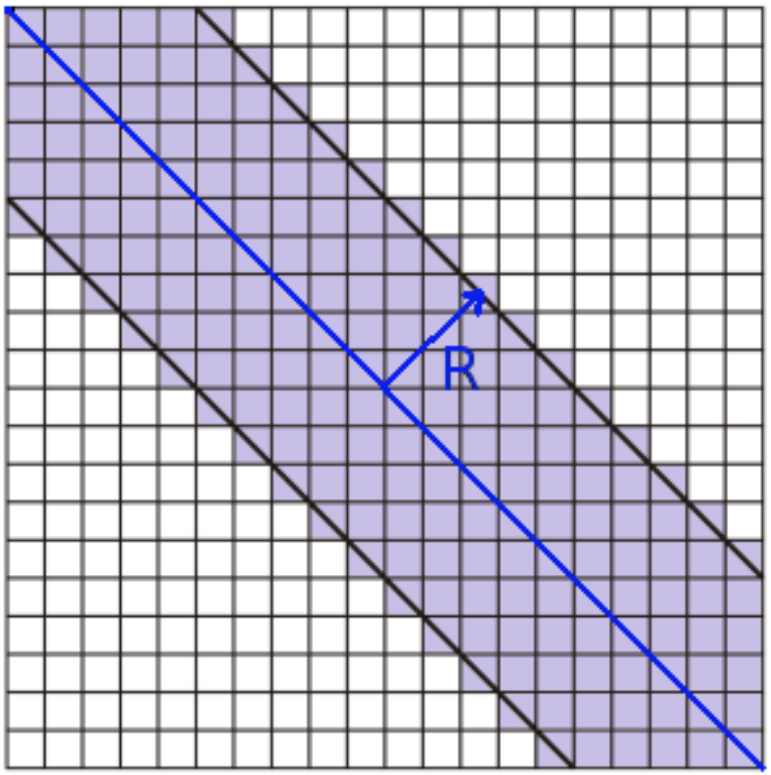
Последовательный алгоритм UCR-DTW

Особенности алгоритма

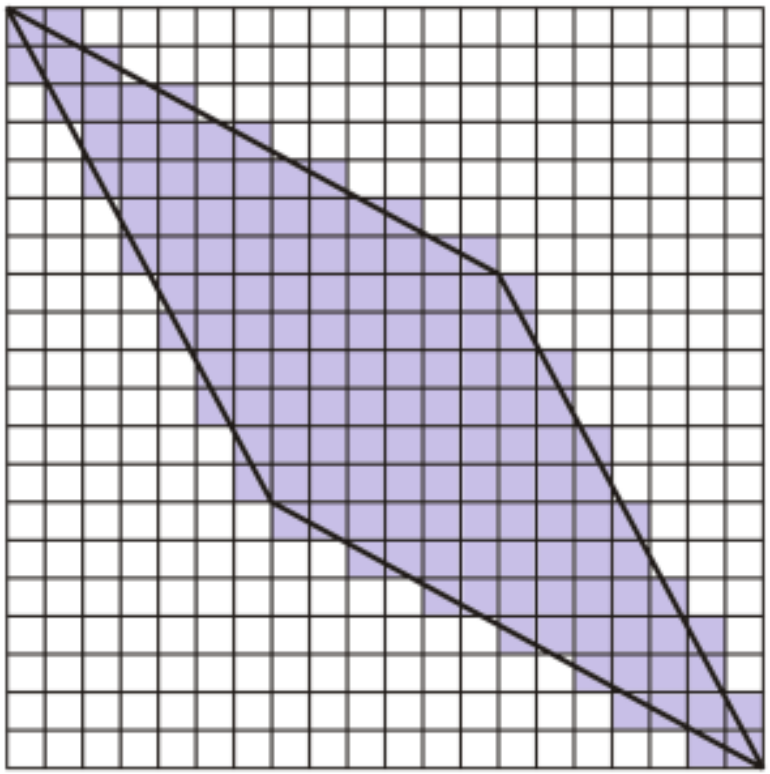
- Возможность использования для поиска в больших временных рядах (т.к. не используются индексы)
- Высокая степень параллелизма
- Высокое быстродействие
- Точный поиск
- Z-нормализация $x'_i = \frac{x_i - \mu}{\sigma}, i \in N$,
  μ – среднее арифметическое, σ – среднеквадратичное отклонение
- DTW в качестве функции схожести

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.

Ограничение DTW



Sakoe-Chiba band



Itakura parallelogram

Оценки DTW

- $LB_{Kim} = \sqrt{(t_0 - q_0)^2 + (t_{n-1} - q_{n-1})^2}$
  Сложность: $O(1)$.
- LB_{Keogh}
  Для запроса Q строятся последовательности U и L.
  $u_i = max(q_{i-R}, q_{i+R})$, $l_i = min(q_{i-R}, q_{i+R})$,

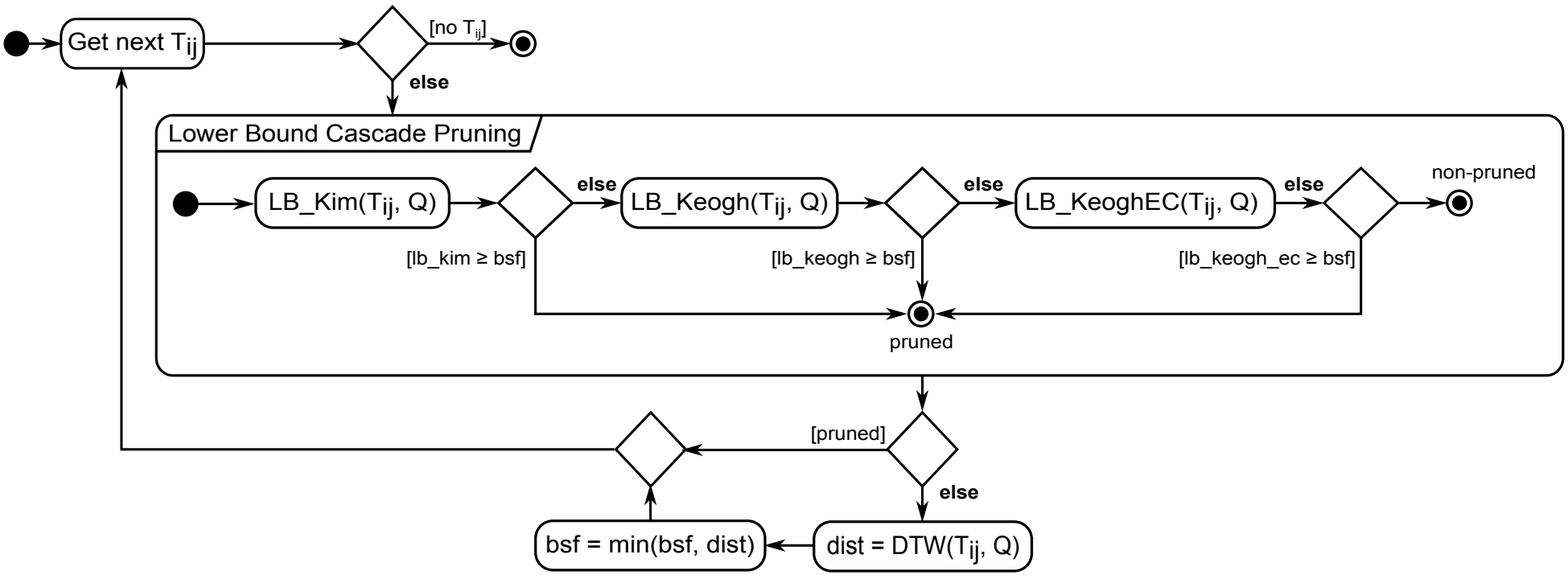
$$LB_{Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (c_i - u_i)^2 & \text{если } c_i > u_i \\ (c_i - l_i)^2 & \text{если } c_i < l_i \\ 0 & \text{в остальных случаях} \end{cases}}$$

  Сложность: $O(n)$.
- $LB_{KeoghEC}$
  Для подпоследовательности C строятся последовательности U и L.
  $u_i = max(c_{i-R}, c_{i+R})$, $l_i = min(c_{i-R}, c_{i+R})$,

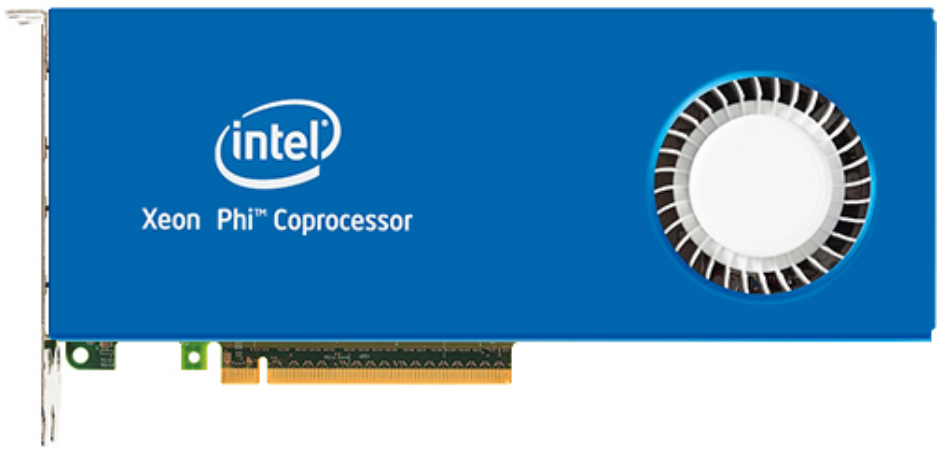
$$LB_{Keogh}(Q, C) = \sqrt{\sum_{i=1}^{n} \begin{cases} (q_i - u_i)^2 & \text{если } q_i > u_i \\ (q_i - l_i)^2 & \text{если } q_i < l_i \\ 0 & \text{в остальных случаях} \end{cases}}$$

  Сложность: $O(n)$.

Последовательный алгоритм UCR-DTW

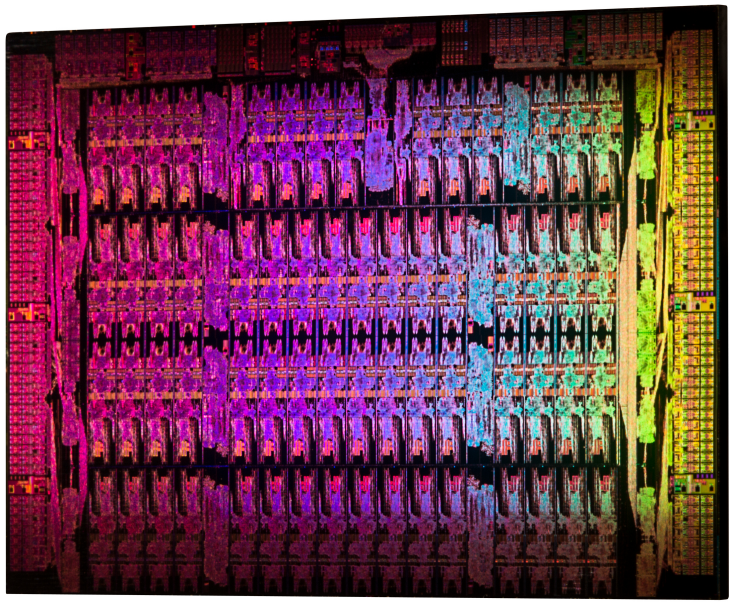


Сопроцесор Intel Xeon Phi



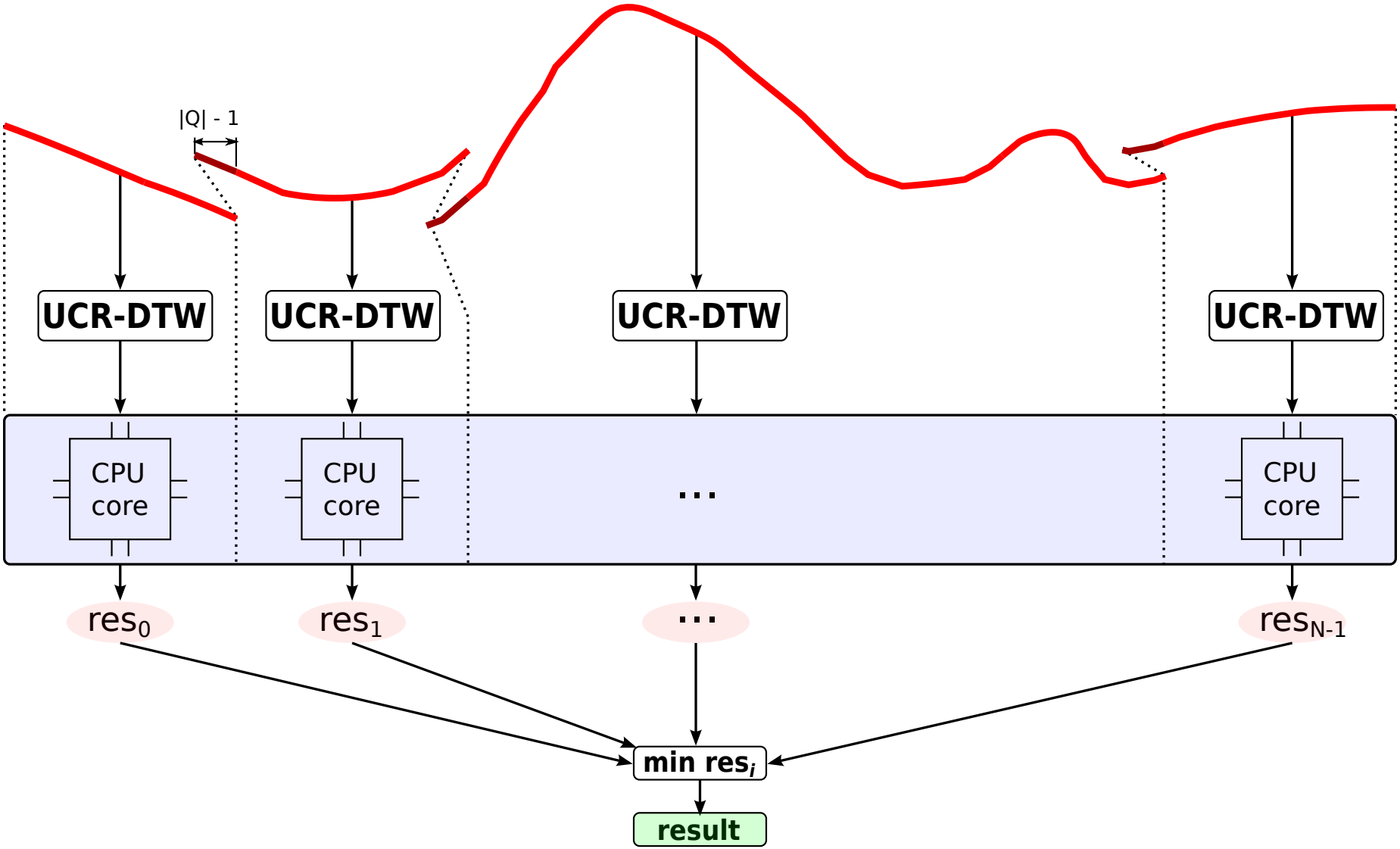
61 ядро, 244 нити, ≈1.2 TFLOPS, 512-bit SIMD

Сопроцессор Intel Xeon Phi

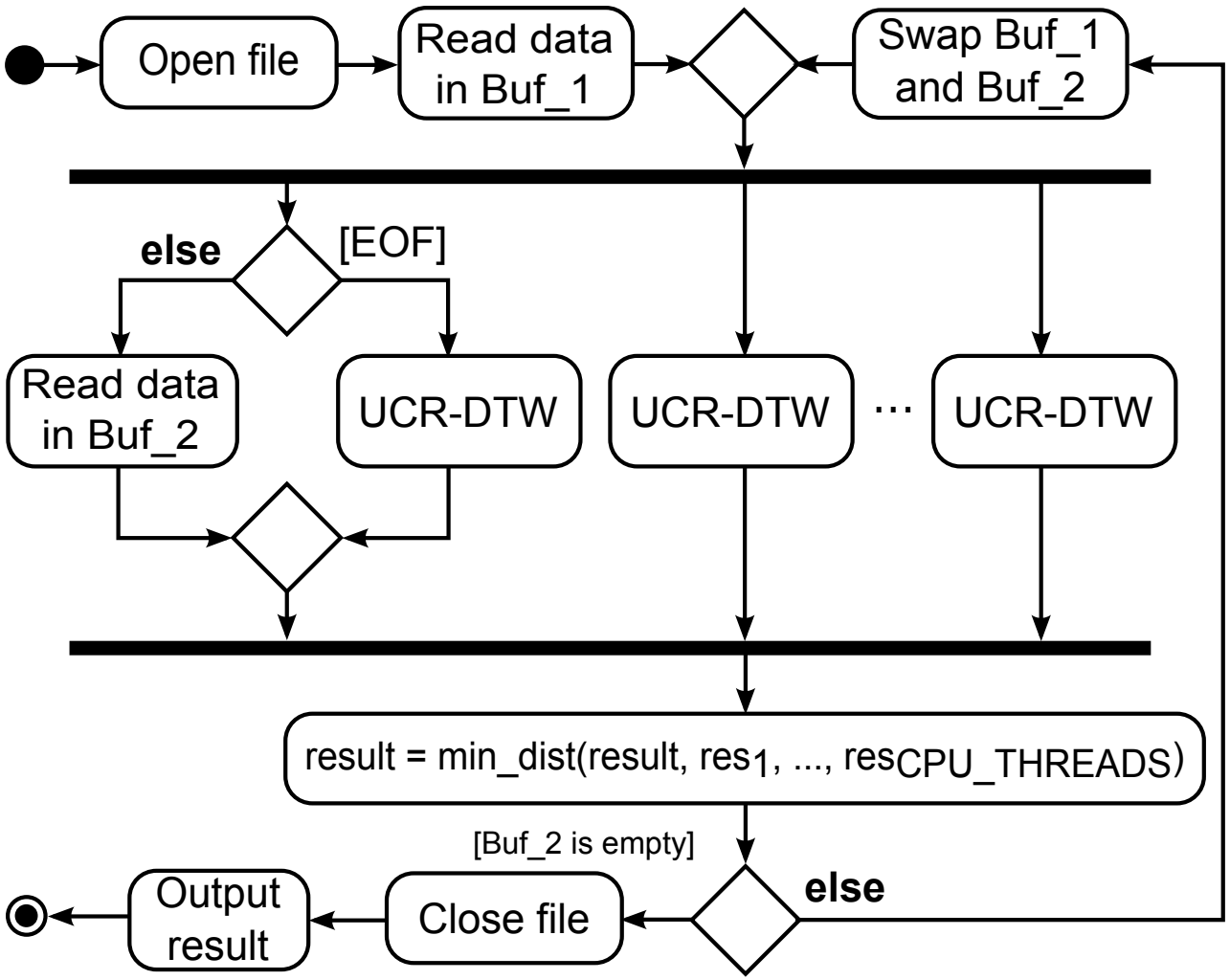


- Выполнение на сопроцессоре (Native Execution)
  - независимое исполнение на сопроцессоре
- Режим выгрузки (Offload)
  - исполнение на процессоре, выгрузка интенсивных вычислений на сопроцессор
- Симметричный режим (Symmetric Mode)
  - исполнение как приложения MPI

Параллельный алгоритм для CPU



Параллельный алгоритм для CPU

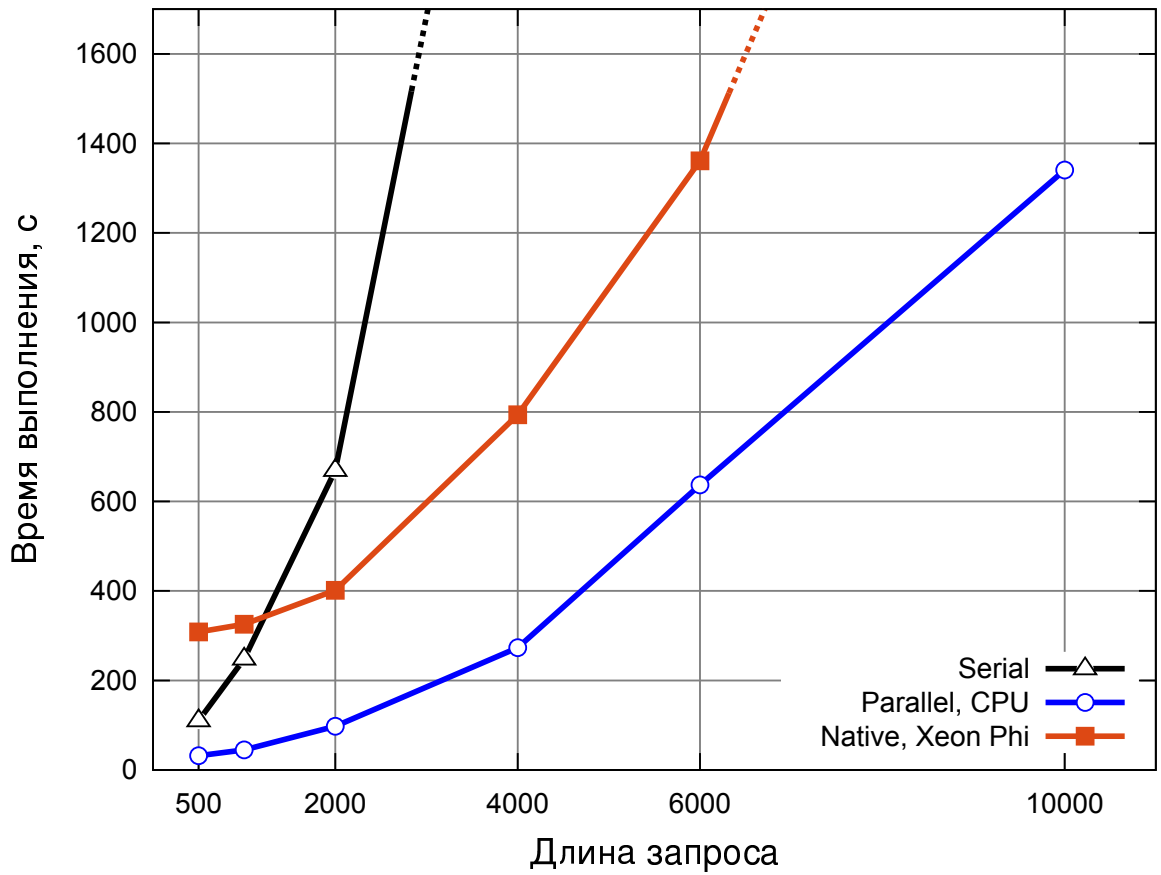


Производительность параллельного алгоритма для CPU

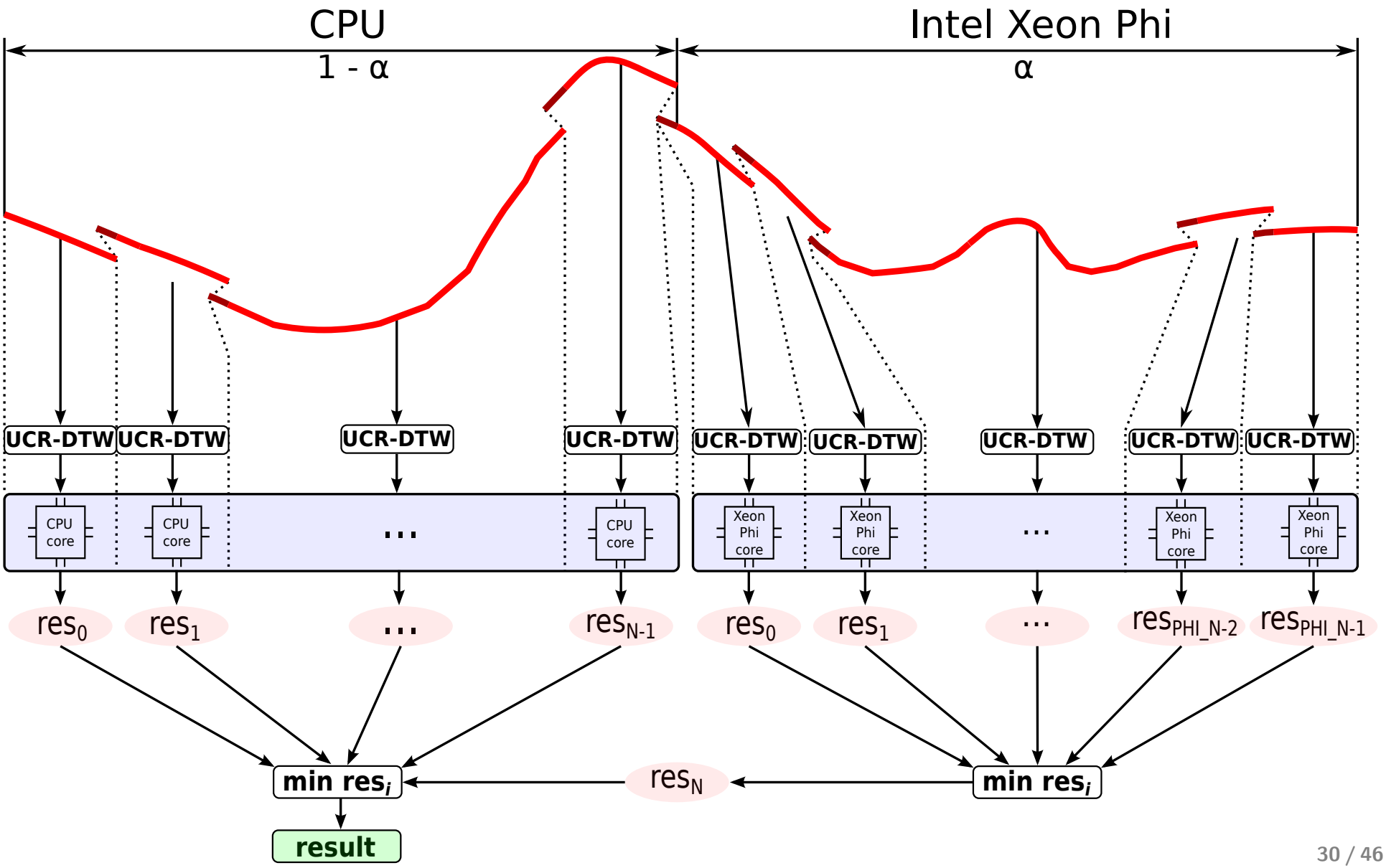
LB_Kim	$O(1)$
LB_Keogh	$O(n)$
LB_KeoghEC	$O(n)$
DTW	$O(n^2)$

Время загрузки данных с диска в память Xeon Phi: $\approx 300\ c$

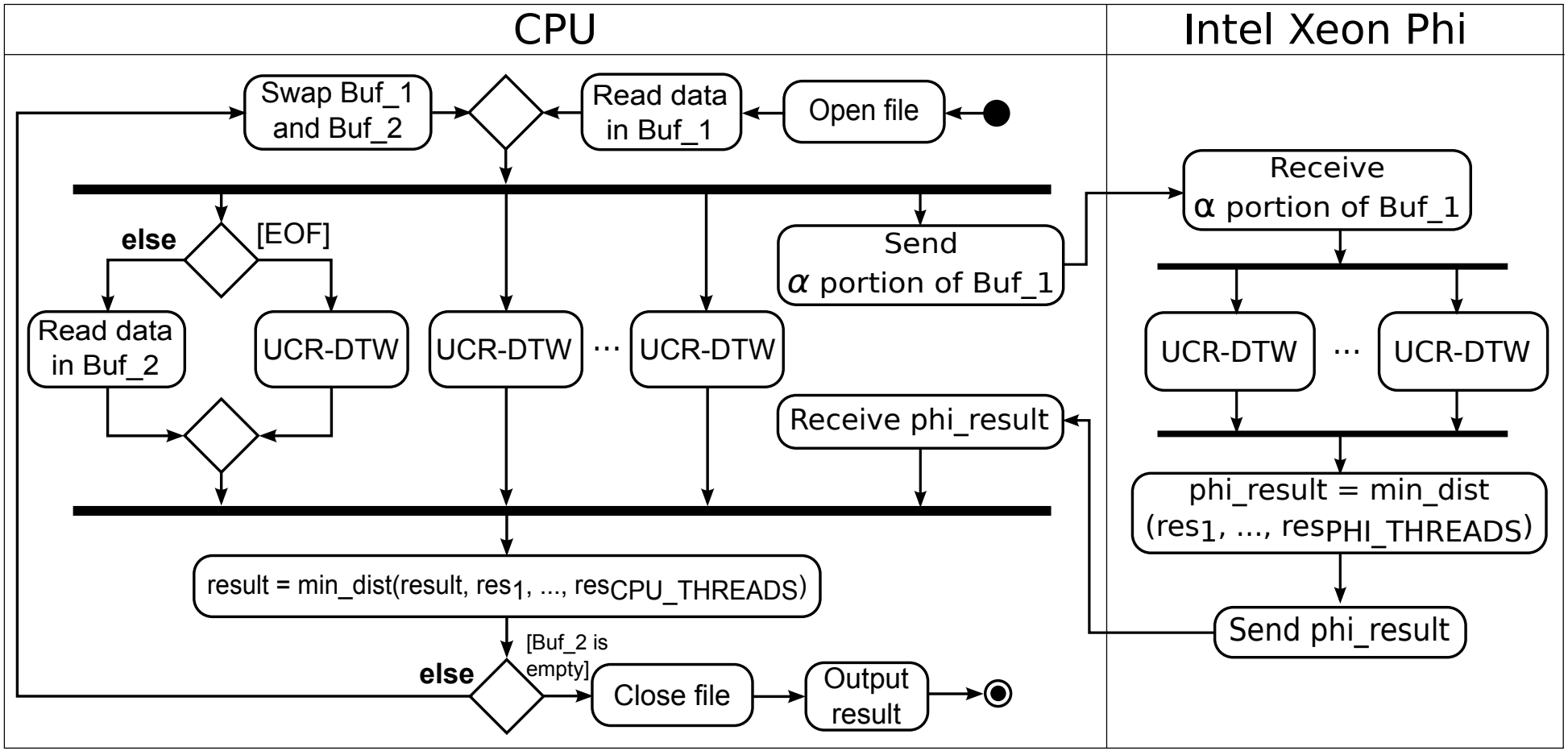
Данные: random walk, 10^9 точек



Наивный параллельный алгоритм для CPU + Xeon Phi

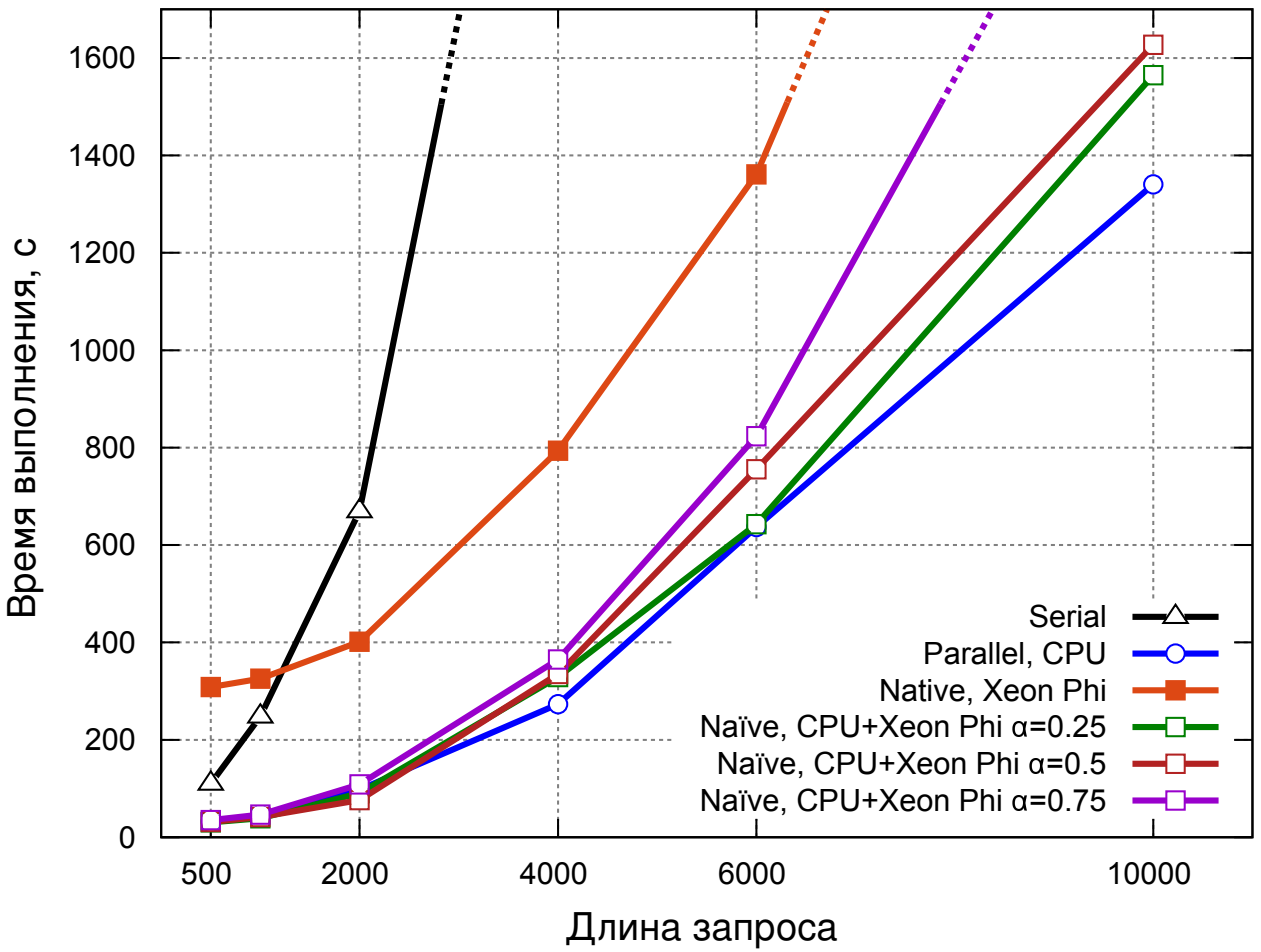


Наивный параллельный алгоритм для CPU + Xeon Phi

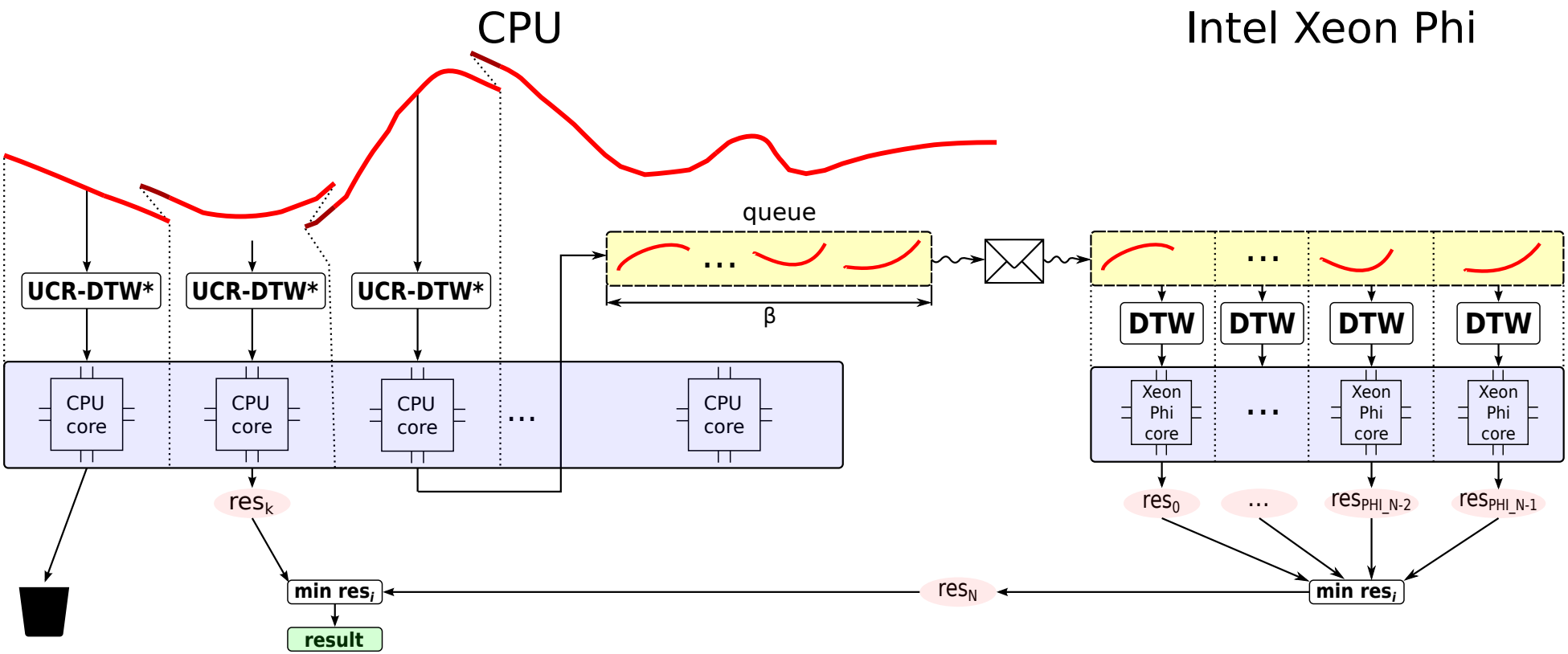


Производительность параллельного алгоритма для CPU + Xeon Phi

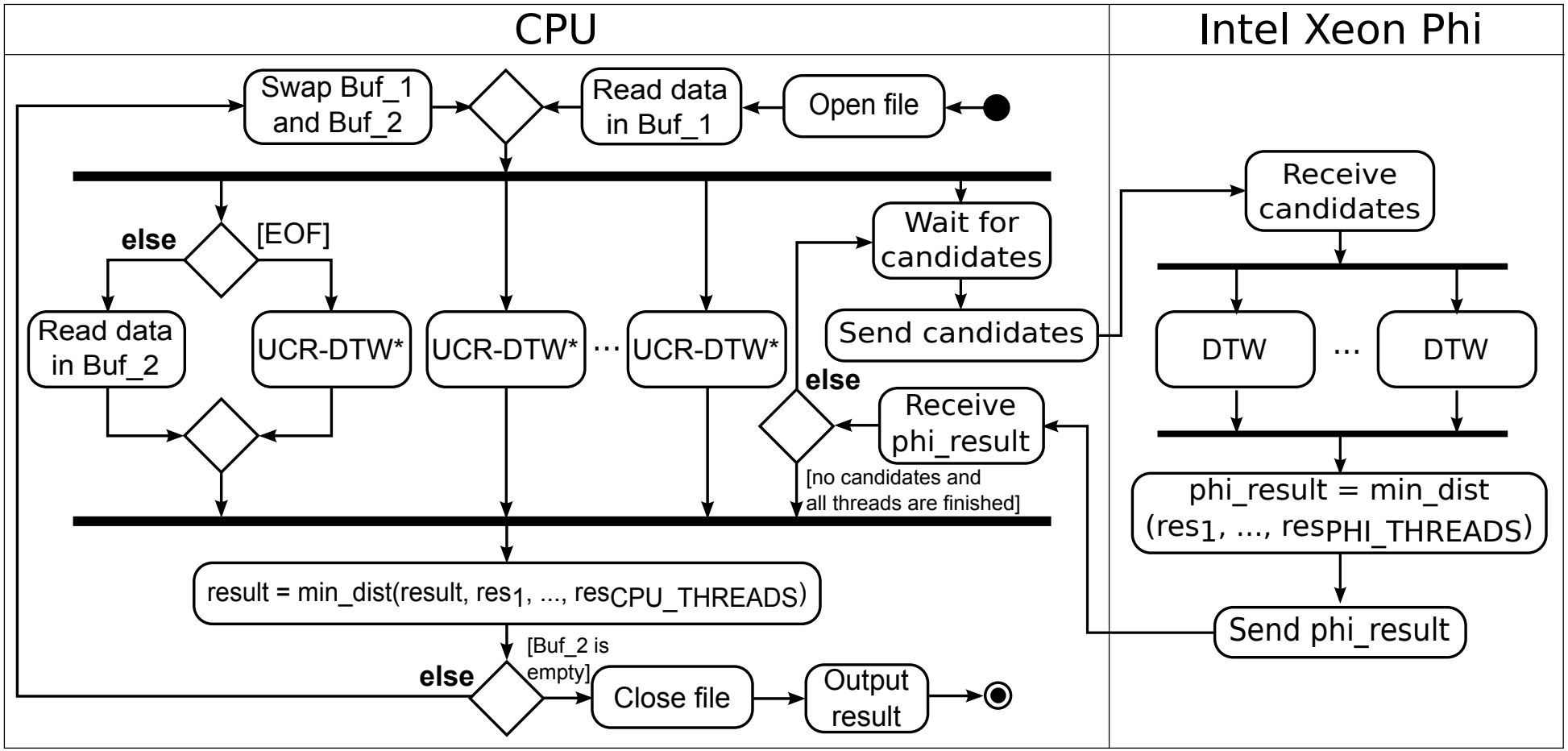
Данные: random walk, 10^9 точек



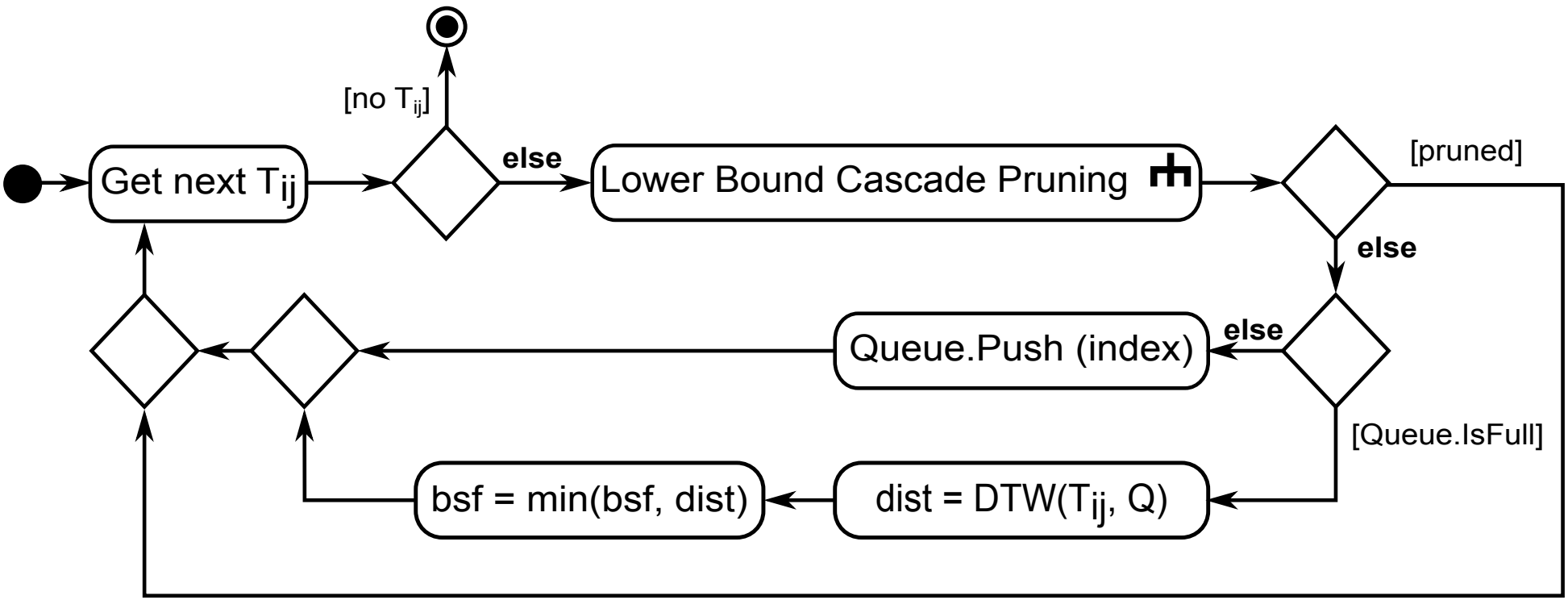
Улучшенный параллельный алгоритм для CPU + Xeon Phi



Улучшенный параллельный алгоритм для CPU + Xeon Phi



UCR-DTW*

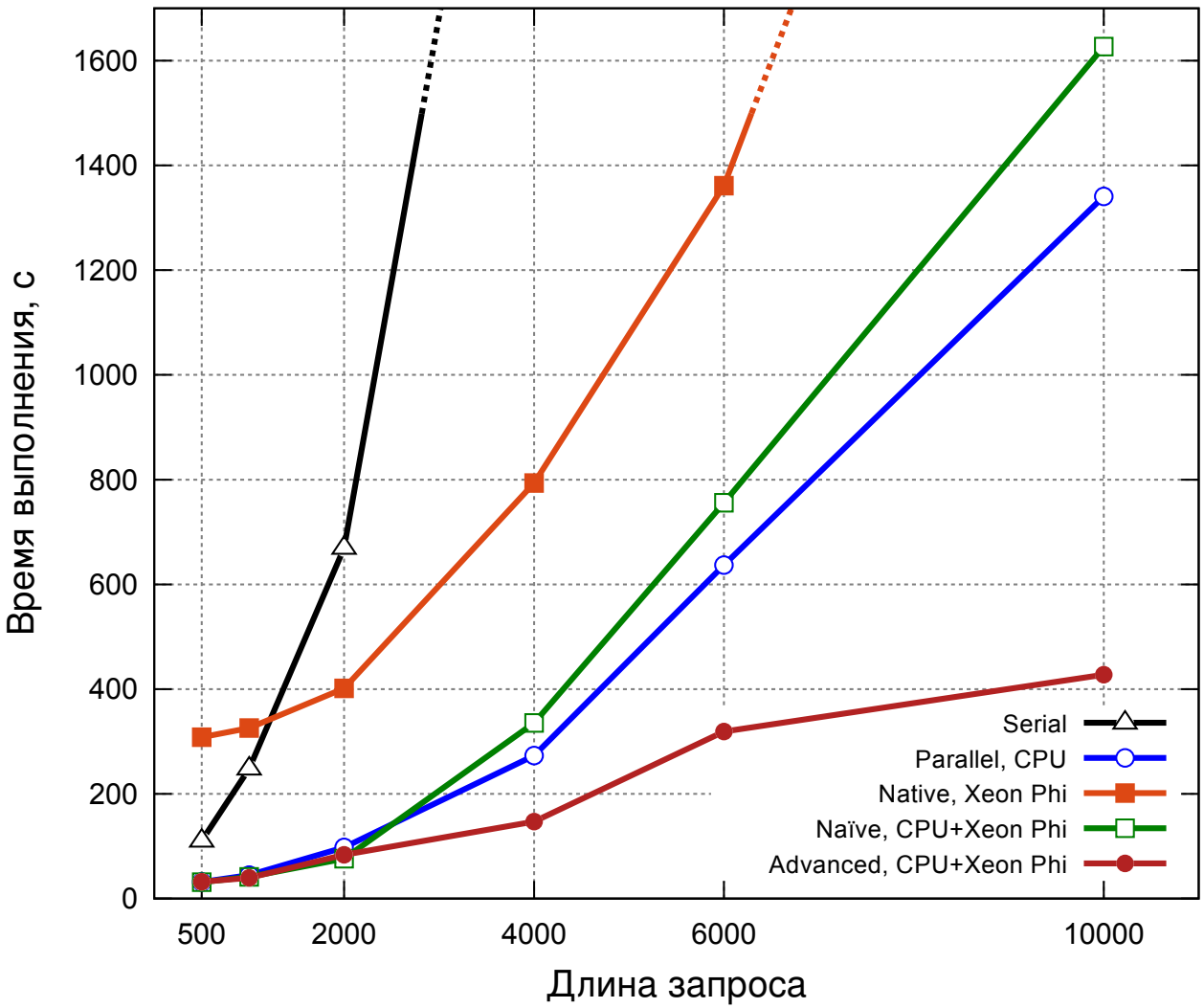

[no T_{ij}]
Get next T_{ij}
else
Lower Bound Cascade Pruning
[pruned]
else
Queue.Push (index)
else
[Queue.IsFull]
bsf = min(bsf, dist)
dist = DTW(T_{ij}, Q)

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, 2×10^7 точек данных (22 час. при частоте дискретизации 250 Гц)

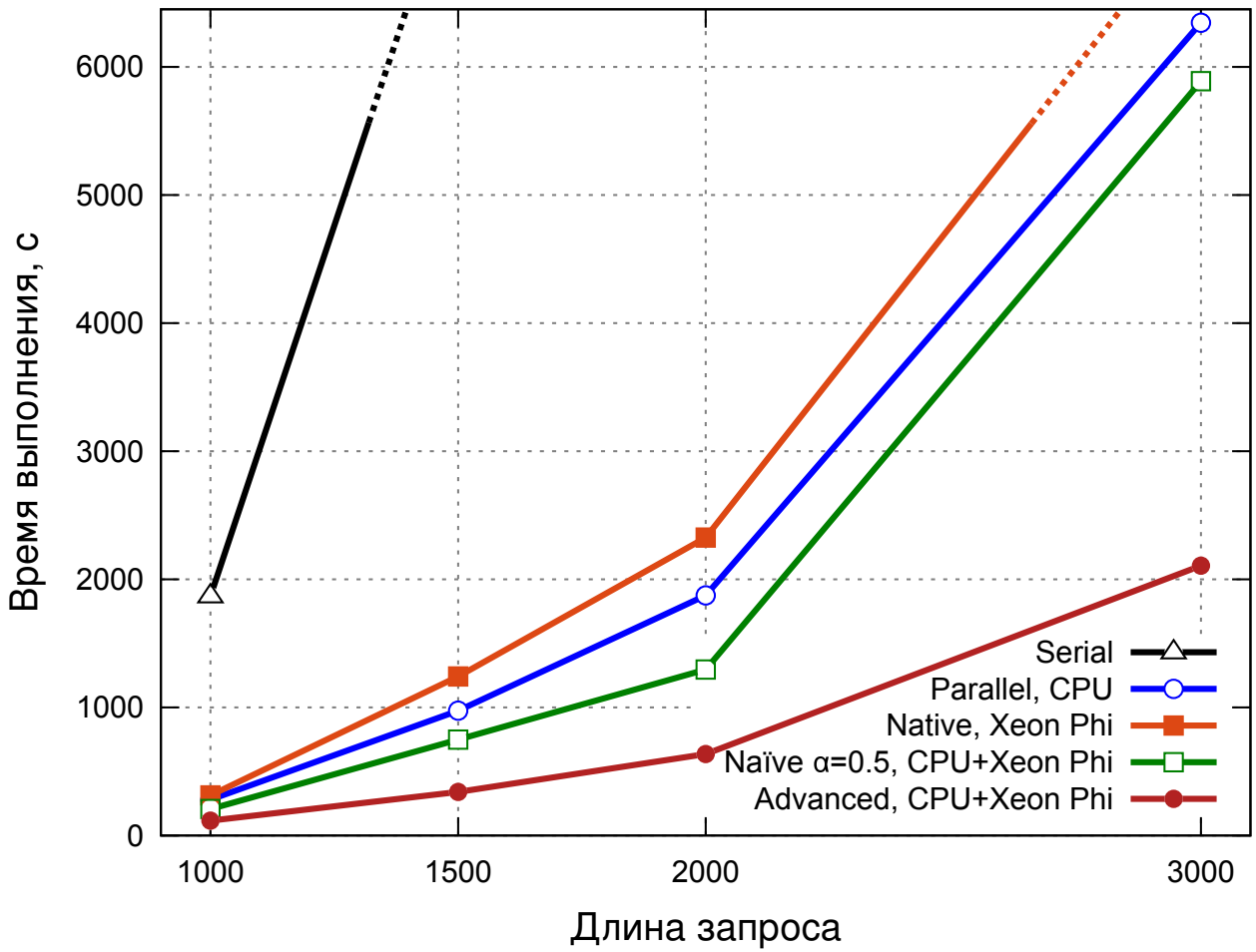
Производительность на синтетических данных

Данные: random walk, 10^9 точек

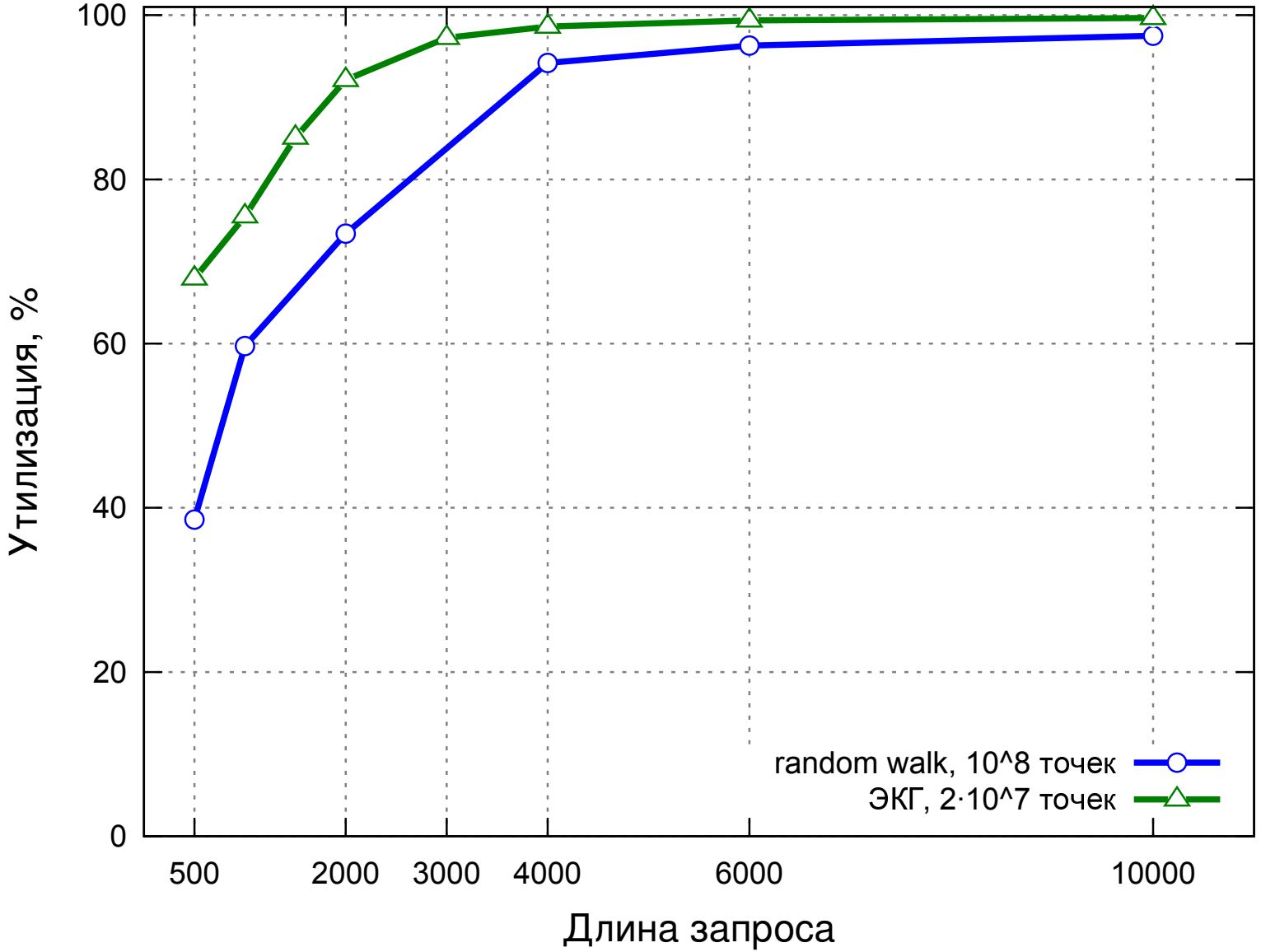


Производительность на реальных данных

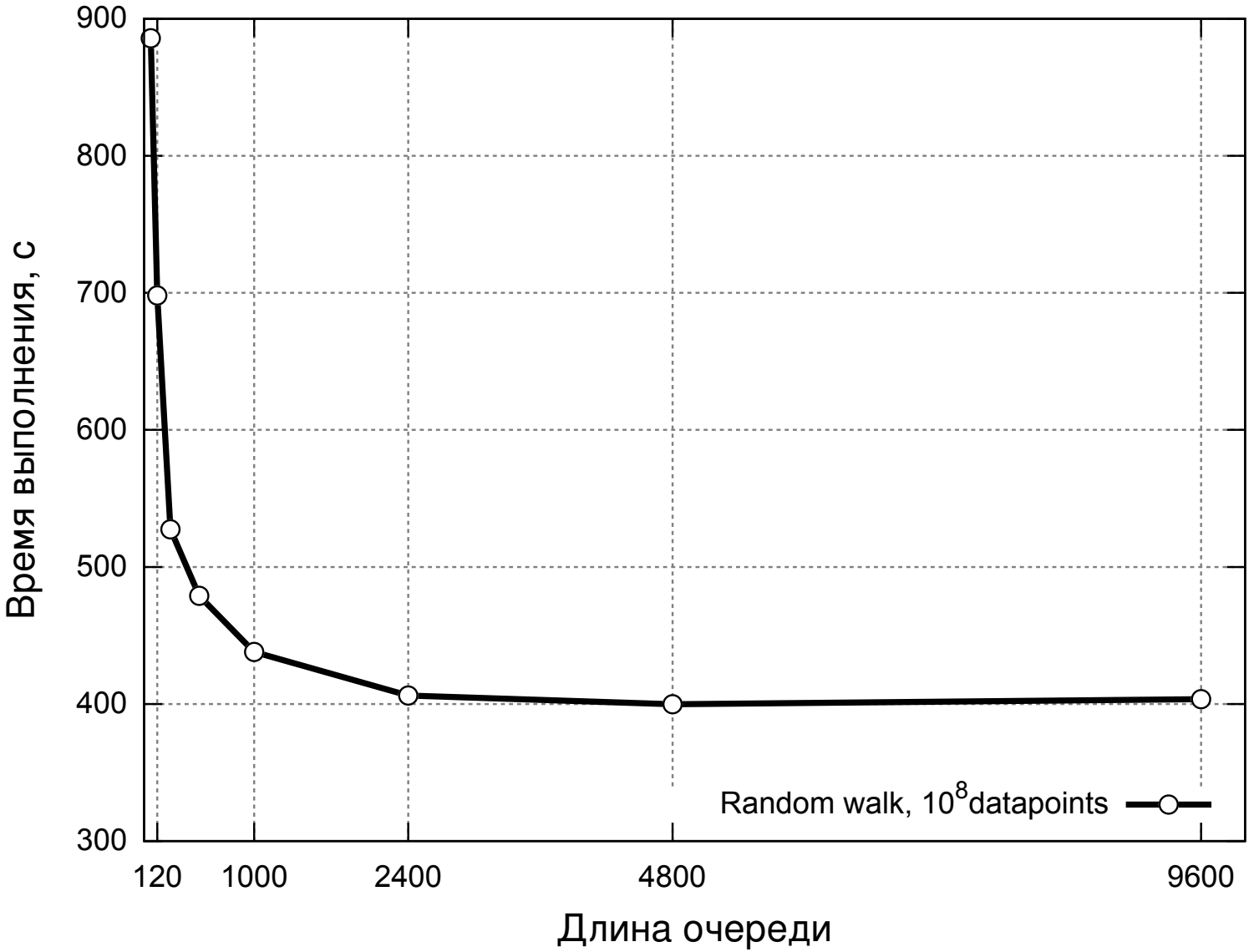
Данные: ЭКГ, 2×10^7 точек



Утилизация сопроцессора



Влияние размера очереди на производительность



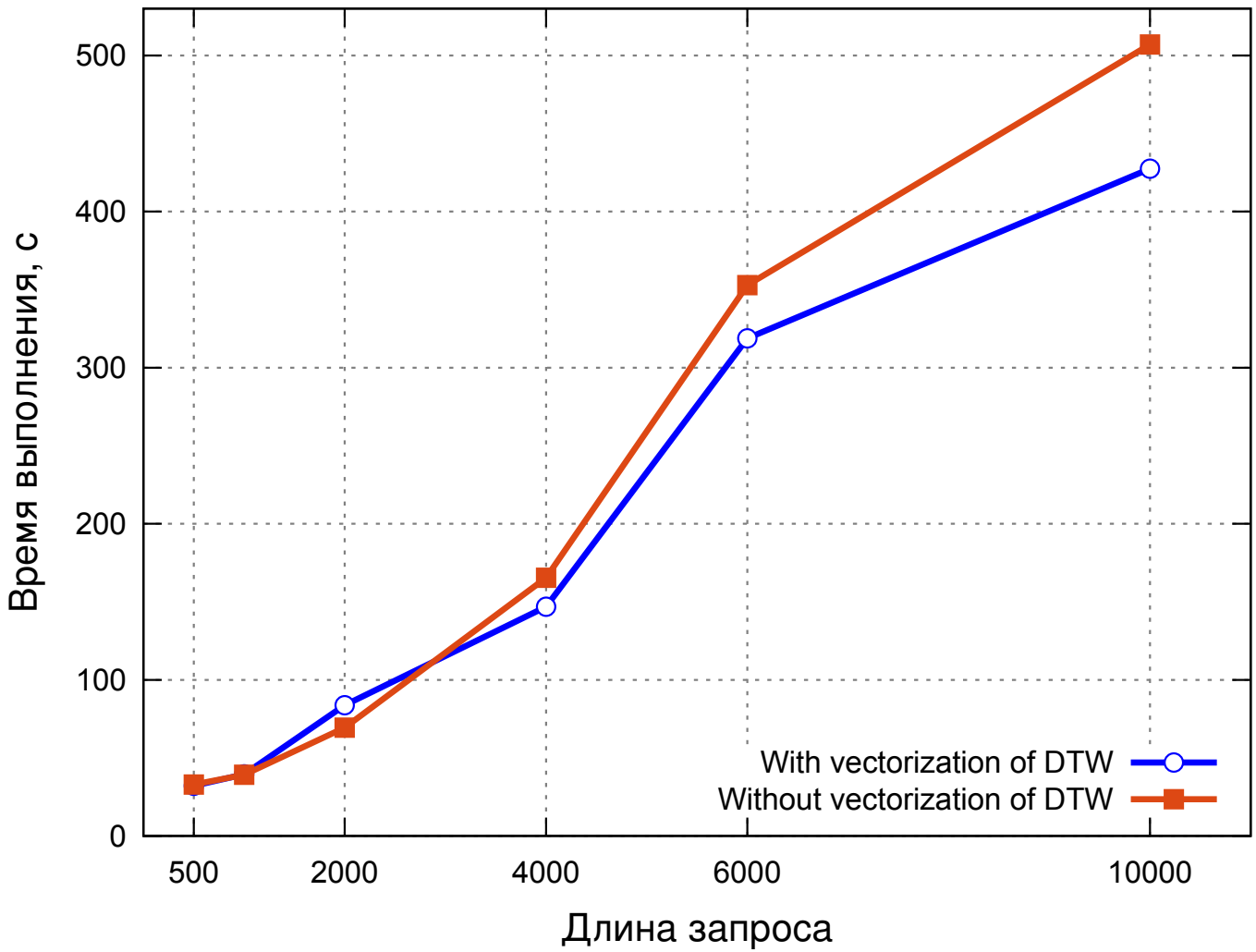
До векторизации DTW

```
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]
}
```

После векторизации DTW

```
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]
}
```

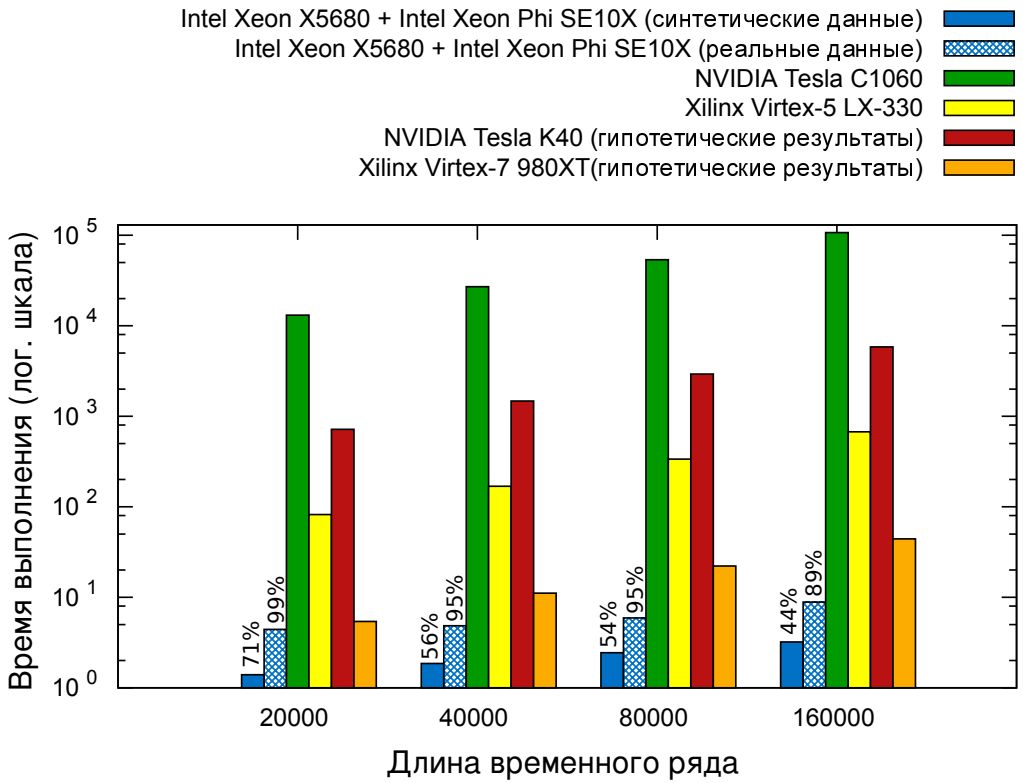
Влияние векторизации DTW на производительность



Сравнение с аналогами (производительность)

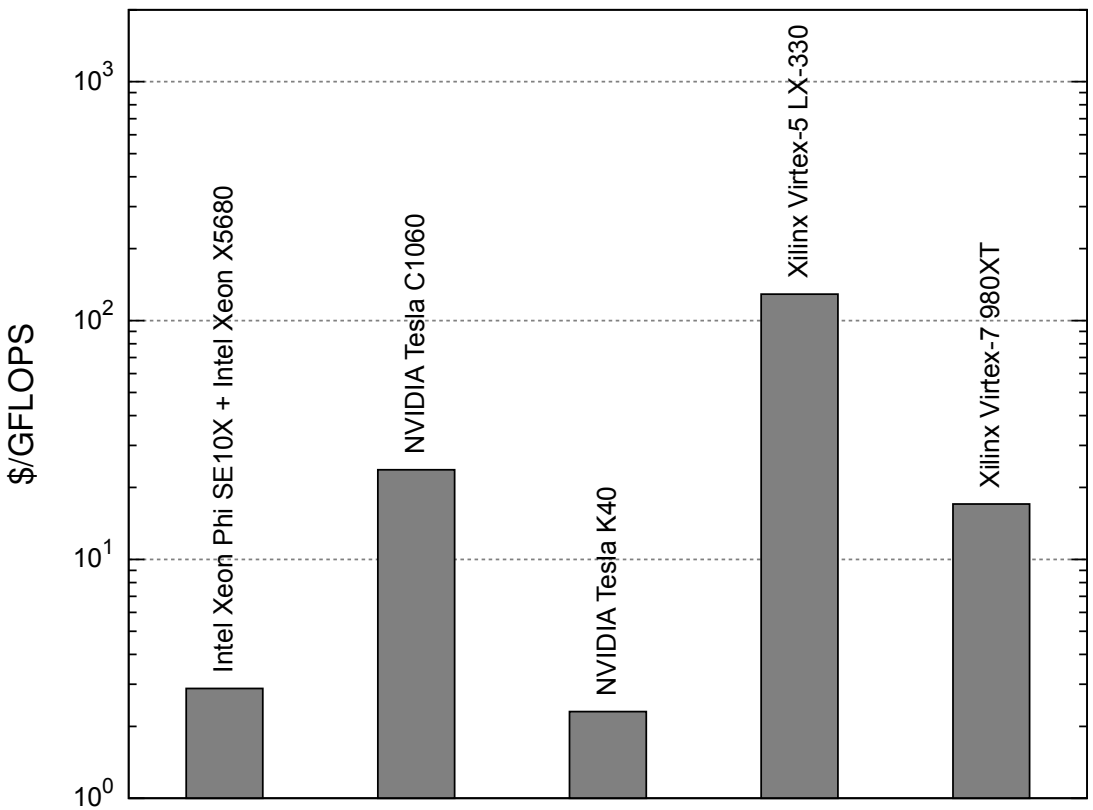
Sart et al. Accelerating dynamic time warping subsequence search with GPUs and FPGAs // ICDM, 2010.
Длина запроса: 1024

Intel Xeon X5680 + Xeon Phi SE10X	1.44 TFLOPS
NVIDIA Tesla C1060	77.8 GFLOPS
Xilinx Virtex-5 LX-330	65 GFLOPS
NVIDIA Tesla K40	1.43 TFLOPS
Xilinx Virtex-7 980XT	0.99 TFLOPS



Сравнение с аналогами (стоимость)

Intel Xeon X5680 + Xeon Phi SE10X	1.44 TFLOPS
NVIDIA Tesla C1060	77.8 GFLOPS
Xilinx Virtex-5 LX-330	65 GFLOPS
NVIDIA Tesla K40	1.43 TFLOPS
Xilinx Virtex-7 980XT	0.99 TFLOPS



Заключение

- Разработан параллельный алгоритм поиска похожей подпоследовательности временного ряда для сопроцессоров Intel Xeon Phi
- Эксперименты показали высокую эффективность алгоритма при большой длине запроса
- Будущие исследования:
  - адаптация для решения задачи local-best-match
  - алгоритм для узла с несколькими сопроцессорами Intel Xeon Phi
  - алгоритм для кластерной системы с узлами на базе сопроцессоров Intel Xeon Phi