Решение задачи разладки на распределенных потоковых данных

Ютман М.А.

Научный руководитель: к.ф.-м.н. Кураленок И.Е.

Санкт-Петербургская школа физико-математических и компьютерных наук

НИУ ВШЭ – Санкт-Петербург

2020

Задача разладки

- Массив из N элементов и семейство $F(\theta)$
- Верно одно из следующих утверждений:
  - Все элементы из $F(\theta_0)$
  - $\exists M, 0 < M < N$, такое что первые M элементов из $F(\theta_0)$, а остальные из $F(\theta_1)$, где $\theta_0 \neq \theta_1$
- Определить, какое верно
- $F(\theta)$ может быть как известно, так и не известно

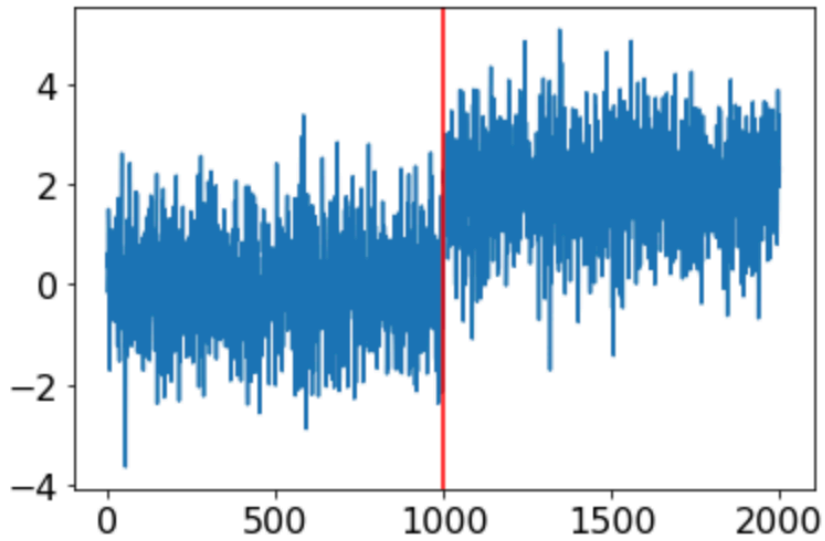


Рис.: Изменение в точке 1000

Потоковая задача разладки

Потоковая версия

- Бесконечный поток элементов
- Изменения в моменты времени $M_1, M_2, \ldots$
- Распределения $F(\theta_0), F(\theta_1), F(\theta_2), \ldots$
- Нужно определить $M_1, M_2, \ldots$

Распределенная версия

- Элементы потока распределяются по вычислительным узлам
- Необходимо задать два алгоритма
  - Алгоритм обработки элемента на узле
  - Алгоритм объединения статистик с разных узлов

Мотивация

Обнаружение спонтанных изменений

- Транзакции кредитных карт
- Тики на фондовой бирже
- Компьютерный сетевой трафик
- Обучающие данные для алгоритмов машинного обучения

Желательно разработать алгоритм

- Распределенный, чтобы увеличить пропускную способность
- Не использующий информацию о входном распределении, так как она не всегда доступна

Предыдущие работы

Решения классической задачи

- Cumulative Sum (CuSum)[1]
- Shiryaev-Roberts procedures (SRP)[2]

Решения потоковой задачи

- Обобщения CuSum и SRP
- Методы со сглаживанием[3]
- Оконный алгоритм[4]

---

[1] Page, E. (1954) 'Continuous inspection schemes', Biometrika, 41, 100-115.

[2] M. Pollak. (1985) 'Optimal Detection of a Change in Distribution', The Annals of Statistics

[3] Bodenham, D.A. (2014). 'Adaptive estimation with change detection for streaming data'. PhD Thesis, Department of Mathematics, Imperial College London.

[4] D. Kifer, S. Ben-David, and J. Gehrke (2004). 'Detecting Change in Data Streams', VLDB.

Ограничение предыдущих работ

Оконный алгоритм[4], в отличие от остальных не использует информацию о семействе распределении данных

Алгоритм	**Инф. о семействе**	**Распред.**
CuSum	Нужна	Нет
SRP	Нужна	Нет
Сглаживание	Нужна	Нет
Оконный	Не нужна	Нет
GLRT	Нужна	Да

НО! Оконный алгоритм не предусмотрен для использования в распределенных системах

[4] D. Kifer, S. Ben-David, and J. Gehrke (2004). 'Detecting Change in Data Streams', VLDB.

Цели и задачи

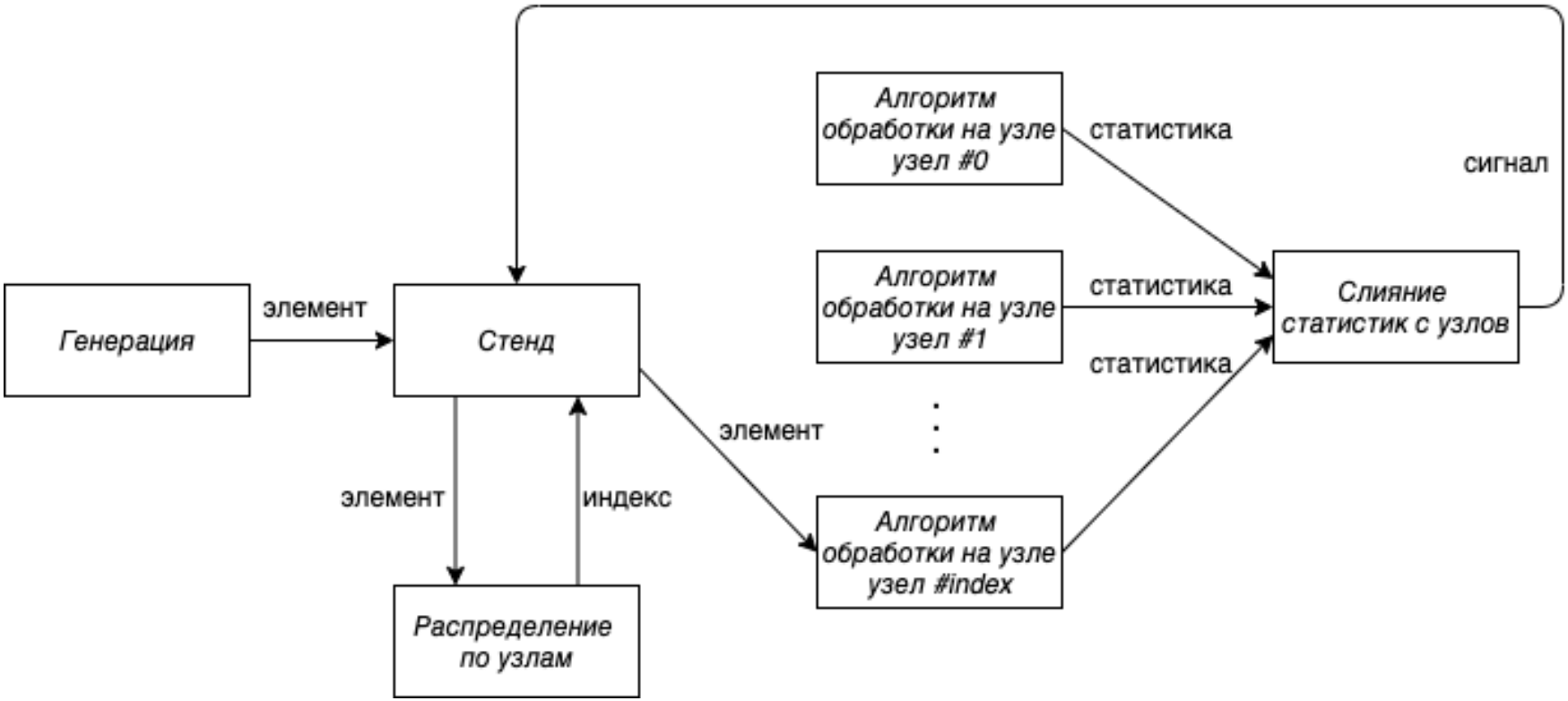
Цель

Обобщить оконный алгоритм обнаружения разладки в потоке на распределенный случай

Задачи

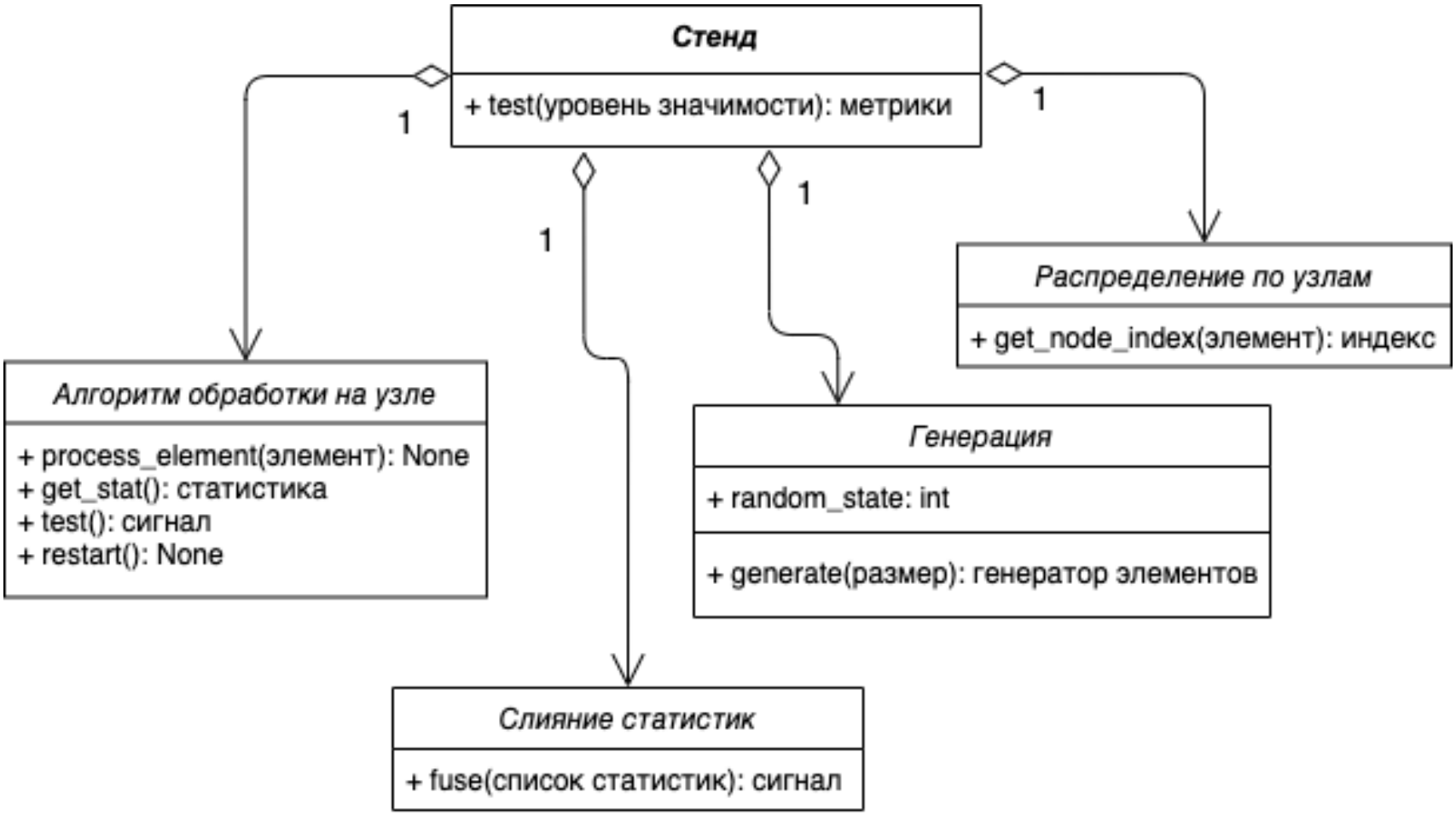
- Создать стенд для тестирования алгоритмов поиска разладки
- Реализовать обобщенный оконный алгоритм в рамках тестового стенда
- Провести серию экспериментов для выявления качества полученного алгоритма
- Реализовать алгоритм в рамках распределенной системы потоковой обработки

Тестовый стенд: архитектура



[5] https://github.com/myutman/distributed-changepoint-detection

Тестовый стенд: интерфейс



[5] https://github.com/myutman/distributed-changepoint-detection

Алгоритм: обработка на узлах

- В качестве статистики с узла возвращается S_t

$$S_t = \max_{1 \le i \le t-(n+m)+1} K(reference, sliding_i)$$

Рис.: Статичное и скользящее окна

  - K — мера близости распределений окон
  - n, m — размеры окон
  - t — число обработанных элементов
- В качестве K используется КС-статистика, $KS(Y_0, Y_1) = \sup |F_{Y_0}(x) - F_{Y_1}(x)|$

Алгоритм: слияние статистик

- К статистикам с узлов применяется функция

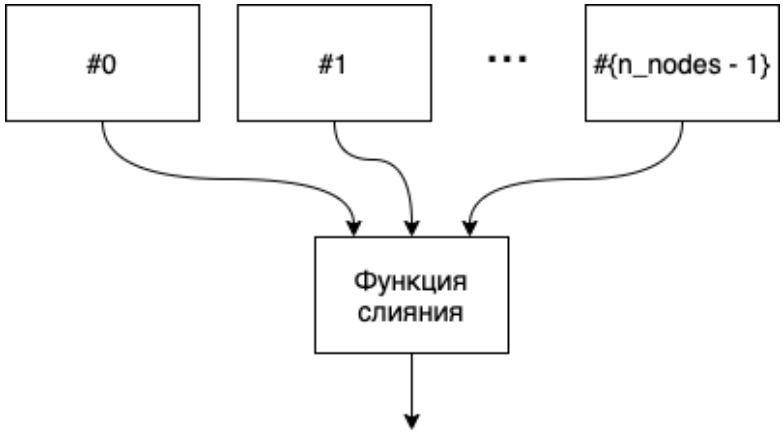


Рис.: Функция слияния

Полученное значение сравнивается с пороговым значением для принятия решения

- Пороговое значение подбирается с учётом **числа узлов** и **уровня значимости** (p)
- В качестве функции слияния использовались **максимум**, **среднее** и **медиана**

Эксперимент без изменений

Поток из $U[0;1]$ без изменений

Алгоритм	**Ложные обнаружения на 2000000 элементов**		
	$p = 0.01$	$p = 0.05$	$p = 0.1$
Ориг. статья	–	8.0	–
Моя реализация			
1 узел	8.6	14.0	22.8
4 узла	3.8	9.6	12.4
8 узлов	1.0	6.6	12.0
16 узлов	1.8	5.6	5.4

- Оригинальные результаты воспроизводятся
- Число ложных обнаружений регулируется p
- Больше узлов $\Rightarrow$ Нужно обработать больше элементов для срабатывания $\Rightarrow$ Число ложных обнаружений убывает

Эксперимент с небольшими изменениями

Поток из $U[-\theta;\theta]$, изначально $\theta := 5$

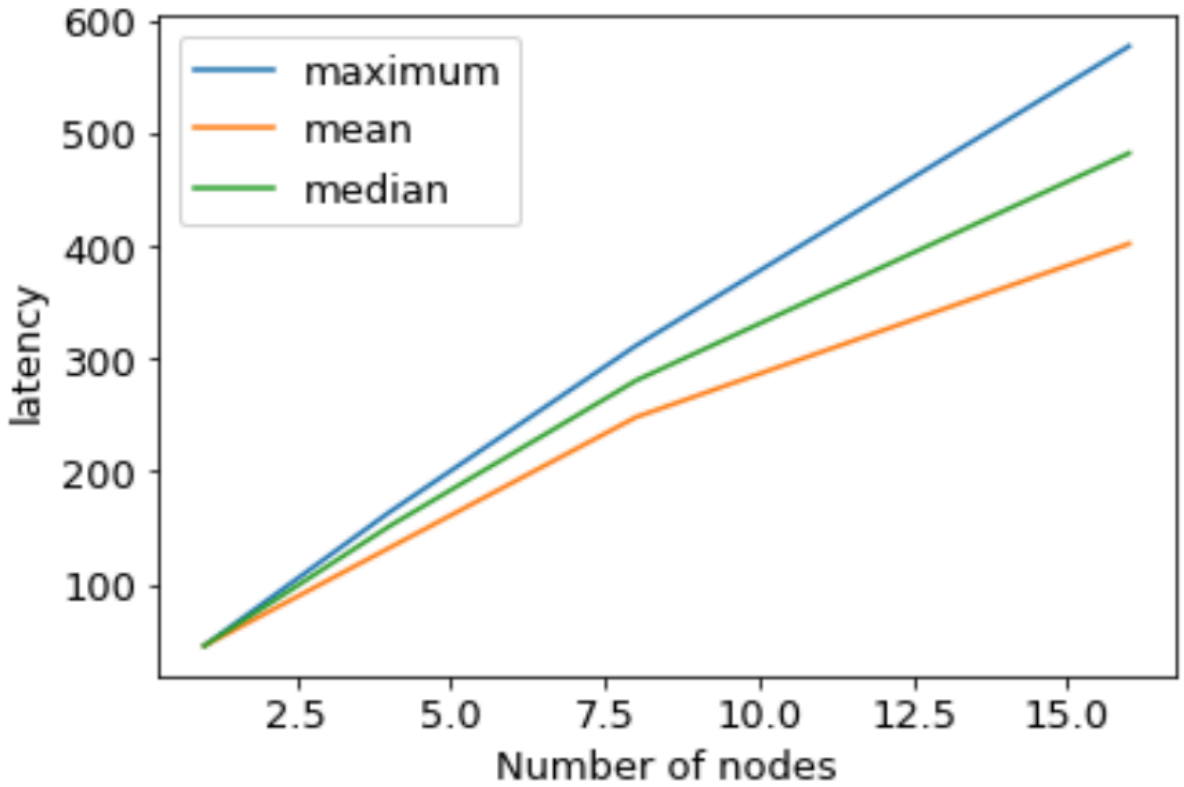
Раз в 20000 элементов $\theta := \theta + U[-1;1]$

Алгоритм	**Число обнаружений на 99 изменений**					
	Верные			**Поздние + Ложные**		
	$p = 0.01$	$p = 0.05$	$p = 0.1$	$p = 0.01$	$p = 0.05$	$p = 0.1$
Ориг. статья	–	31	–	–	30	–
Моя реализация						
1 узел	24.0	33.4	42.0	$18.4 + 2.0$	$18.8 + 7.2$	$17.2 + 15.0$
4 узла	24.8	31.6	37.4	$18.2 + 1.0$	$20.8 + 3.4$	$20.4 + 8.4$
8 узлов	21.0	25.6	32.4	$19.2 + 0.2$	$21.4 + 1.2$	$23.6 + 2.4$
16 узлов	13.6	18.8	22.4	$20.0 + 0.4$	$21.2 + 0.8$	$20.8 + 0.6$

- Оригинальные результаты воспроизводятся
- Больше узлов $\Rightarrow$ Нужно обработать больше элементов для срабатывания
  - $\Rightarrow$ Общее число обнаружений убывает
  - $\Rightarrow$ Процент поздних обнаружений возрастает

Эксперимент с большими изменениями

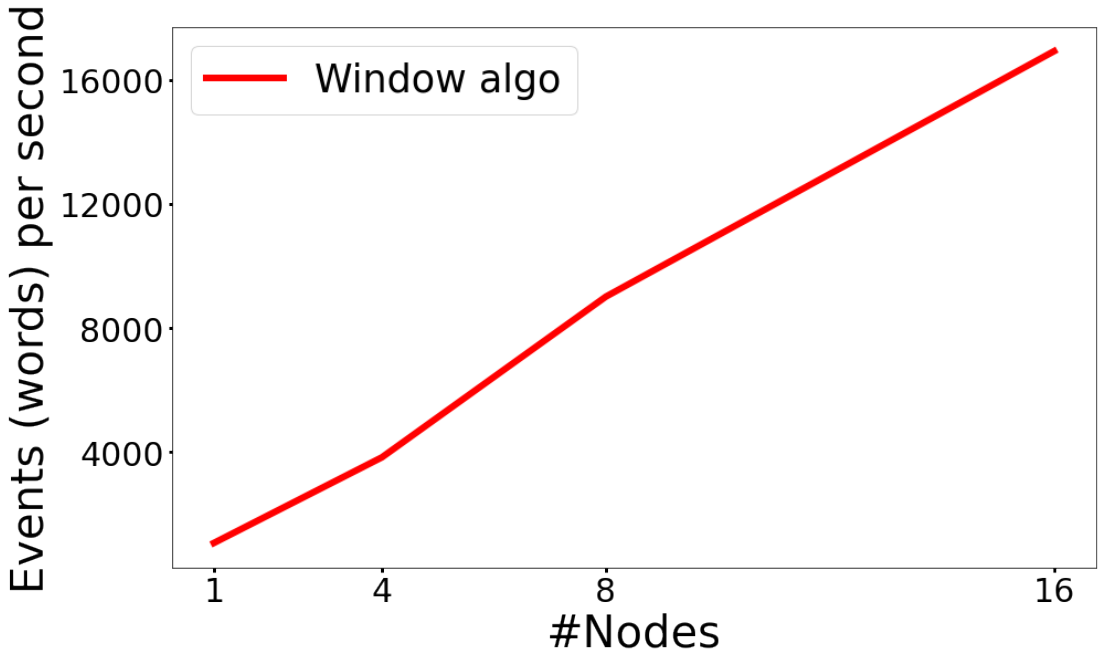
- Поток размера 400000 из $N(\theta, 1)$
- Изначально $\theta = 0$
- Раз в $N(20000, 1)$ элем. θ меняется на $N(100, 0.1)$



- Задержка растёт сублинейно
- Среднее даёт наименьшую задержку

Распределенная система

- Алгоритм был реализован в рамках распределенной системы Apache Flink[6]
- Были произведены измерения пропускной способности



- Пропускная способность возрастает линейно с увеличением числа узлов

[6]https://github.com/flame-stream/LightBulbs/pull/1

Результаты

- Был создан тестовый стенд
- Был разработан распределенный алгоритм для потоковой задачи разладки, не использующий информацию о распределении входных данных
- Результаты оригинальной работы были воспроизведены
- С увеличением числа узлов
  - Общее число и число ложных обнаружений убывают
  - Процент поздних обнаружений возрастает
  - $latency$ возрастает сублинейно
  - $throughput$ возрастает линейно
- Была продемонстрирована масштабируемость
- По результатам планируется публикация